Volume 19 Number 2

ACTA CYBERNETICA

Szeged, 2009

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. Fifty reprints are supplied for each article published.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above informations along with the contents of past issues are available at the Acta Cybernetica homepage http://www.inf.u-szeged.hu/actacybernetica/ .

Bálint Dömölki
IQSOFT
Budapest, Hungary
domolki@iqsoft.hu

Zoltán Ésik
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
ze@inf.u-szeged.hu

Zoltán Fülöp
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
fulop@inf.u-szeged.hu

Ferenc Gécseg
Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
gecseg@inf.u-szeged.hu

Jozef Gruska
Institute of Informatics/Mathematics
Slovak Academy of Science
Bratislava, Slovakia
gruska@savba.sk

Tibor Gyimóthy
Department of Software Engineering
University of Szeged
Szeged, Hungary
gyimothy@inf.u-szeged.hu

Helmut Jürgensen
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
helmut@csd.uwo.ca

Alice Kelemenová
Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun
Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa
Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa
Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga
Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
vogler@inf.tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Weighted Automata: Theory and Applications

Guest Editors:

Manfred Droste
Institut für Informatik
Universität Leipzig
Germany
droste@informatik.uni-leipzig.de

Heiko Vogler
Fakultät Informatik
Technische Universität Dresden
Germany
Heiko.Vogler@tu-dresden.de

Preface

This special issue contains papers on topics of the workshop

"Weighted Automata: Theory and Applications (WATA 2008)"

which took place at the Technische Universität Dresden, Germany, May 13-16, 2008.

As for its predecessors WATA 2002, WATA 2004, and WATA 2006, the goal of this workshop was to highlight the field of weighted automata, ranging from the theory of formal power series to applications of tree automata, natural language processing, XML, and multi-valued logics.

The workshop was attended by 46 participants from 12 countries. Two tutorials were given by

Z. Ésik (Szeged, Hungary and Tarragona, Spain)
K. Knight (Los Angeles, USA).

In addition, seven invited lectures were presented by

F. Drewes (Umeå, Sweden),
B. Gerla (Varese, Italy),
A. Maletti (Berkeley, USA),
G. Rahonis (Thessaloniki, Greece)
S. Gaubert (Rocquencourt, France),
W. Kuich (Vienna, Austria),
W. Martens (Dortmund, Germany),

Furthermore, 18 talks were selected as contributed communications.

This workshop was financially supported by the "Gesellschaft von Freunden und Förderern der TU Dresden" and the "International Center for Computational Logic".

After the workshop, a call for papers for a special issue of "Acta Cybernetica" on "Weighted Automata: Theory and Applications" was issued. Following the standard refereeing procedure, we were pleased to accept the present seven papers for this special issue.

Manfred Droste (Leipzig)
Heiko Vogler (Dresden)

September 2009

Acta Cybernetica 19 (2009) 249–274.

MAT Learners for Recognizable Tree Languages and Tree Series

Frank Drewes*

Abstract

We review a family of closely related query learning algorithms for unweighted and weighted tree automata, all of which are based on adaptations of the minimal adequate teacher (MAT) model by Angluin. Rather than presenting new results, the goal is to discuss these algorithms in sufficient detail to make their similarities and differences transparent to the reader interested in grammatical inference of tree automata.

Keywords: algorithmic learning, grammatical inference, tree automaton, tree language, tree series

1 Introduction

This article discusses a family of algorithms for grammatical inference of unweighted and weighted tree automata. Traditionally, the area of grammatical inference studies the problem of learning a formal (string) language L by automatically inferring an explicit automata-theoretic or grammatical description A of L from examples or some other type of information about L. In other words, the aim is to come up with a *learner*, an algorithm that exploits a source S of information about L in order to construct A. Different so-called learning models are obtained by specifying (a) which source S of information the learner is provided with, (b) how the learner gets access to this information, and (c) what the exact criterion of success is.

The three most well-established categories of learning models in grammatical inference are Gold's *learning from examples with identification in the limit* [23], Valiants *probably approximately correct (PAC) learning* [39], and Angluin's *query learning* [4].

Here, we focus on query learning. This model, which is also called active learning, gives the learner access to a *teacher*, an oracle able to answer certain types of queries. Suppose that L is a regular string language and the goal is to construct a corresponding finite-state automaton A. The most well-studied type of teacher is the so-called minimal adequate teacher (MAT) [3]. The MAT will answer two different sorts of queries regarding L. The first is the *membership query*, in which

*Umeå University, 906 87 Umeå, Sweden, E-mail: drewes@cs.umu.se

the learner passes the teacher a string u, and the teacher checks whether $u \in L$. In the second type of query, the *equivalence query*, the learner passes the teacher a proposed automaton A', and the teacher checks whether A' correctly describes L. If so, A' is accepted and the learning process terminates. Otherwise, the teacher returns a counterexample to the learner, i.e., an element of the symmetric difference of L and the language described by A'.

A learning model closely related to MAT learning is learning from representative samples and membership queries [2]. Here, the learner has access to a weaker teacher who will only answer membership queries. To compensate for the lack of equivalence queries, the learner is initially provided with a *representative sample*, a set of strings in L, such that every transition of A is used at least once when processing the strings in the sample.

Here, we want to consider algorithms for learning unweighted and weighted tree automata rather than ordinary finite-state automata. Why would such extensions be of interest? Apart from theoretical curiosity and the fact that tree languages play an important role in many application areas, motivation is provided by the fact that almost all results regarding the inference of context-free languages are negative. However, recognizable (or regular) tree languages may be seen as context-free languages whose strings are enriched with explicit structural information. Thus, positive results for grammatical inference of recognizable tree languages make it possible to learn context-free languages if the learner is provided with the additional structural information (cf. [32]).

If we want to use the learning models described above, they have to be adapted. This can be done in a straightforward way. In membership queries, trees rather than strings must be passed to the teacher, and in equivalence queries, tree automata of the type considered must be checked by the teacher. Similarly, a representative sample is now a set of trees. Moreover, in the weighted case, membership queries must be replaced with coefficient queries (i.e., the teacher returns the coefficient of the tree passed, with respect to the sought tree series), and the counterexample returned as an answer to an equivalence query must be a tree for which the proposed automaton computes a coefficient that differs from the one it should compute.

The appropriateness of the MAT model is not undisputed. Obviously, the assumption of having access to an oracle able to answer equivalence queries is strong and may be considered unrealistic. Moreover, it has been argued in [6] that membership queries are oversimplified and should be replaced by a type of query yielding a more informative result, e.g., so-called correction queries. To a certain extent, this criticism is certainly justified. In particular, future research should continue to explore reasonable alternative settings. However, in the author's opinion, this does not diminish the value of the algorithms reviewed in the next two sections. In general, one should keep in mind that the learning models considered are idealizations that – as always in Theoretical Computer Science – trade realism for mathematical elegance and simplicity. Having read this paper, the reader who has never seen these algorithms before will hopefully acknowledge that they are based on beautiful formal reasonings. In particular, they make elegant use of Myhill-Nerode-like characterizations of the tree languages and series to be learned.

Since grammatical inference is an inherently difficult goal, there seem to be only two ways to achieve positive results whose correctness can formally be proved. One either has to simplify the goal, e.g., by placing severe restrictions on the concepts to be learned, or give the learner access to a rather powerful source of information, such as a MAT. Clearly, both approaches have their advantages and disadvantages. This paper focuses on the second, because we are interested in the grammatical inference of unrestricted recognizable tree languages and tree series. For this task, there do not yet seem to exist many algorithms other than the ones discussed here. Moreover, these algorithms are all very closely related to each other, which makes them interesting (in the authors opinion), because it indicates that they are based on "robust" ideas worth being explored.

As mentioned above, the MAT model is a formal idealization. Therefore, one cannot expect that learning algorithms based on a formal setting such as the MAT model can directly be applied to learning tasks in, say, natural language processing. However, it may be an interesting goal to pursue in future research to identify practical scenarios in which the teacher can be simulated by, e.g., statistical methods. Of course, such an approach would no more be guaranteed to yield an affirmatively correct answer, but it may perform sufficiently well in practice – and hopefully much better than an ad-hoc approach. In fact, it may then be a theoretically interesting and practically well-motivated question under which assumptions imperfect teachers give rise to reasonably good results, e.g., in a PAC-like setting.

From what has been said above, it should be clear that this paper is not a general survey of the large field of grammatical inference. In fact, it does not even attempt to cover the subarea of grammatical inference of tree languages and tree series. Readers who wish to obtain a general overview of grammatical inference are referred to the various existing survey papers [1, 13, 21, 28, 34]. Readers interested in inference of tree languages, using other methods and models than the ones discussed here, may also wish to have a look at [33, 27, 20, 29].

In the next section, learners for recognizable tree languages based on (variations of) the MAT model are discussed. In Section 3, we discuss generalizations of these algorithms, that learn recognizable tree series. The paper concludes with some final remarks in Section 4.

2 Learners for Recognizable Tree Languages

As mentioned above, grammatical inference is the task to construct an automaton or a grammar describing a language L, given certain information about L. For the moment being, let us consider the string case. Suppose that we are interested in learning a class $\mathcal{L}(\mathcal{A})$ of string languages, where $\mathcal{A}$ is a class of automata, and $\mathcal{L}(\mathcal{A}) = \{L(A) \mid A \in \mathcal{A}\}$ is the class of languages generated by $\mathcal{A}$. The task of the learner is to construct, for a given language $L \in \mathcal{L}(\mathcal{A})$, an automaton $A \in \mathcal{A}$ with $L(A) = L$. For this, the learner needs to have access to information regarding L. Here, we mainly want to study the case where this information is provided by a MAT [3]. This oracle that will (correctly) answer two different sorts of queries:

Membership query Given a string $u \in \Sigma^*$ (provided by the learner), the membership query member(u) will be answered by returning 1 if $u \in L$, and 0 if $u \notin L$. (Thus, member computes the characteristic function of L; see below.)

Equivalence query Given an automaton $A \in \mathcal{A}$ (provided by the learner), the equivalence query eqQuery(A) will be answered by returning the special token $\perp$ if $L(A) = L$. Otherwise, a counterexample $u \in L(A) \triangle L$ is returned, where the operator $\triangle$ yields the symmetric difference of sets.

The learner L_* proposed in [3] learns the class of regular languages from a MAT in polynomial time, where $\mathcal{A}$ is the set of total deterministic finite-state automata. It makes use of the Myhill-Nerode theorem for regular languages to construct the canonical finite-state automaton recognizing L.[1] To achieve this goal, the learner maintains a so-called observation table, which can be seen as an adapted version of the *state characterization matrix* introduced by Gold [24] for identifying regular languages from positive and negative examples in the limit. In the following, we discuss extensions and variations of L_* that learn tree automata.

Let us first recall a few basic definitions and facts. A ranked alphabet Σ is a finite set of ranked symbols (f, k), where f is a symbol and $k \in \mathbb{N}$, its rank, is a non-negative integer. We let $\Sigma_{(k)} = \{(f, l) \in \Sigma \mid l = k\}$. In the following, a ranked symbol (f, k) will simply be denoted by f, or by $f^{(k)}$ if it is necessary to specify its rank. The set T_Σ of trees over Σ is the smallest set of formal expressions such that $f[t_1, \ldots, t_k] \in T_\Sigma$, for every $f^{(k)} \in \Sigma$ ($k \in \mathbb{N}$) and all $t_1, \ldots, t_k \in T_\Sigma$. Here, the brackets and commas are special symbols not in Σ. For $k = 0$, the tree $f[]$ may simply be denoted by f. For a set T of trees, we let $\Sigma(T)$ denote the set of all trees of the form $f[t_1, \ldots, t_k]$, where $f^{(k)} \in \Sigma$ and $t_1, \ldots, t_k \in T$. The set of all subtrees of a tree $t = f[t_1, \ldots, t_k]$ is given by $\text{subtrees}(t) = \{t\} \cup \bigcup_{i=1}^{k} \text{subtrees}(t_i)$. A *tree language* is a set $L \subseteq T_\Sigma$. The characteristic function of L is denoted by χ_L. Thus, for $t \in T_\Sigma$, $\chi_L(t) = 1$ if $t \in L$, and $\chi_L(t) = 0$, otherwise.

Definition 2.1. *A deterministic bottom-up finite tree automaton (fta) is a tuple $A = (\Sigma, Q, \delta, F)$ consisting of a ranked alphabet Σ, a ranked alphabet Q of states such that $Q = Q_{(0)}$, a transition table δ, and a set $F \subseteq Q$ of final states. The transition table is a partial function $\delta\colon \Sigma(Q) \to Q$. This extends to trees in the canonical way, yielding a partial function $\delta\colon T_\Sigma \to Q$. A tree $t \in T_\Sigma$ is accepted by A if $\delta(t) \in F$. The language recognized by A consists of all trees accepted by A, i.e., $L(A) = \{t \in T_\Sigma \mid \delta(t) \in F\}$, and is called a recognizable (or regular) tree language.*

As usual, an fta is said to be total if the transition table δ is a total function. We note that δ can also be regarded as a set of transitions $f[q_1, \ldots, q_k] \to q$, where $\delta(f[q_1, \ldots, q_k]) = q$. In other words, a transition is a pair in $\Sigma(Q) \times Q$. Since we consider only the deterministic case, transitions have pairwise distinct left-hand sides $f[q_1, \ldots, q_k]$. However, unless the fta is total, not all left-hand sides need to be present.

[1] It may be interesting to note that the class of regular languages is not learnable in polynomial time from membership or equivalence queries alone [5]. This provides some justification for calling the oracle above a *minimal adequate teacher*.

The first extension of L_* to so-called skeletal tree languages[2] was given by Sakakibara [32]. Let us have a look at this learner, which we may call L_*^{tfta}. It constructs the canonical total fta recognizing the target language L. In the presentation below, we drop the restriction to skeletal tree languages, since it is not important for the correctness of L_*^{tfta}. In fact, this slight generalization has the advantage that L_* may be seen as a special case of L_*^{tfta}, by identifying a string $a_1 \cdots a_n$ with the monadic tree $a_n[\cdots a_1[\epsilon]\cdots]$. (The string case can, of course, even be simulated using skeletal trees, but this seems to require the use of a representation that maps strings to trees in a non-surjective way, for example, by representing $u = a_1 \cdots a_n$ as $\mathrm{tree}(u) = *[\cdots *\,[a_1, a_2], \cdots a_n]$. As a consequence, if A is a deterministic finite-state string automaton, an fta recognizing $\{\mathrm{tree}(u) \mid u \in L(A)\}$ will in general contain more states than A.)

As indicated in the introduction, the idea behind L_* and all its descendants is to construct an automaton by exploiting the Myhill-Nerode congruence of the target language. Let $\Box^{(0)} \notin \Sigma$ be a special symbol, and let C_Σ be the set of all trees in $T_{\Sigma\cup\{\Box\}}$ with exactly one occurrence of $\Box$, called *contexts over* Σ. The concatenation $c \cdot t$ of $c \in C_\Sigma$ with $t \in T_\Sigma \cup C_\Sigma$ is the tree obtained from c by replacing $\Box$ with t. Now, the Myhill-Nerode congruence $\equiv_L$ on T_Σ is given by

$$t \equiv_L t' \text{ if and only if } \chi_L(c \cdot t) = \chi_L(c \cdot t') \text{ for all } c \in C_\Sigma.$$

It is well known that $\equiv_L$ is of finite index (i.e., its congruence classes are finite in number) if and only if L is recognizable. The canonical (total) fta A_L^{t} recognizing L can be obtained as usual, by taking the congruence classes $[t]_{\equiv_L}$, $t \in T_\Sigma$, as states and defining $\delta(f[[t_1]_{\equiv_L}, \ldots, [t_k]_{\equiv_L}]) = [f[t_1, \ldots, t_k]]_{\equiv_L}$. By the congruence property, the choice of the representatives $t_1, \ldots, t_k$ does not matter. A state $[t]_{\equiv_L}$ is final if $t \in L$.

Now, let us define an equivalence relation $\sim_C$ on T_Σ by replacing C_Σ in the definition of $\equiv_L$ with a finite set of contexts. For $C \subseteq C_\Sigma$, let $t \sim_C t'$ if and only if, for all $c \in C$, $\chi_L(c \cdot t) = \chi_L(c \cdot t')$. By definition, $\equiv_L = \sim_{C_\Sigma}$. Moreover, if $\equiv_L$ is of finite index, there is a finite set C of contexts such that $\equiv_L = \sim_C$. The learners based on L_* (and, in fact, several other learners as well), discover such a set C and construct the target automaton from it. Note that, for arbitrary $C \subseteq C_\Sigma$, $\sim_C$ is not necessarily a congruence.

Following the same idea as L_*, the learner L_*^{tfta} uses membership and equivalence queries to discover trees representing different congruence classes, together with suitable separating contexts. The data structure used for this is the previously mentioned observation table. Its rows are indexed by the trees in $\Sigma(S)$, for a finite set $S \subseteq T_\Sigma$, and its columns are indexed by contexts from a finite set $C \subseteq C_\Sigma$ containing $\Box$. The cell in row t and column c contains the value $\chi_L(c \cdot t)$, which the learner obtains by asking a membership query. For $t \in \Sigma(S)$, if the observation table Ω in question is clear from the context, we let $\langle t \rangle$ denote the C-indexed vector given by the row of t in Ω. For a set $T \subseteq \Sigma(S)$, we let $\langle T \rangle = \{\langle t \rangle \mid t \in T\}$.

[2] A tree language L is *skeletal* if $L \subseteq T_\Sigma$ for a ranked alphabet Σ with $|\Sigma_{(k)}| \leq 1$ for all $k \geq 1$.

We require that S be *subtree-closed*, meaning that $s_1, \ldots, s_k \in S$ for every tree $f[s_1, \ldots, s_k] \in S$. In other words, $S \subseteq \Sigma(S)$, which means that Ω even contains rows for the trees $s \in S$. Note that, for $t, t' \in \Sigma(S)$, $\langle t \rangle \neq \langle t' \rangle$ implies $t \not\equiv_L t'$, because $\sim_C \supseteq \equiv_L$. Moreover, as observed above, there exists an observation table for which the converse holds as well. The aim of the learner is to build such an observation table.

During its run, the learner $\mathrm{L}_*^{\mathrm{tfta}}$ repeatedly uses the tentative observation table Ω it has built in order to construct a total fta A_Ω consistent with the observations in Ω. This fta is passed to the teacher, and if it is not approved, then the counterexample received is used to enlarge Ω. To be able to construct A_Ω from Ω, the following two properties are needed.

1. Ω is *closed*, meaning that $\langle t \rangle \in \langle S \rangle$, for every $t \in \Sigma(S)$.
2. Ω is *consistent*. To define this property, let $\Sigma_\square(S) = \mathrm{C}_\Sigma \cap \Sigma(S \cup \{\square\})$. The observation table Ω is consistent if $\langle c \cdot s \rangle = \langle c \cdot s' \rangle$, for all $c \in \Sigma_\square(S)$ and all $s, s' \in S$ with $\langle s \rangle = \langle s' \rangle$. Note that $\langle c \cdot s \rangle \neq \langle c \cdot s' \rangle$ would mean that there is a $d \in C$ such that $\chi_L((d \cdot c) \cdot s) \neq \chi_L((d \cdot c) \cdot s')$, i.e., $d \cdot c$ would be a context witnessing that $s \not\equiv_L s'$, despite the fact that $\langle s \rangle = \langle s' \rangle$. Moreover, the addition of $d \cdot c$ to C would make the rows of s and s' different, thus resolving the inconsistency.

If Ω is both closed and consistent, A_Ω can be defined by a construction similar to the construction of the canonical fta from $\equiv_L$. The set of states is $\langle S \rangle$, a state $\langle s \rangle$ being final if $s \in L$. For every tree $t = f[s_1, \ldots, s_k] \in \Sigma(S)$, we let $\delta(f[\langle s_1 \rangle, \ldots, \langle s_k \rangle]) = \langle t \rangle$. Note that, by the closedness of Ω, $\langle t \rangle$ belongs to $\langle S \rangle$. Consistency is needed to ensure that $\delta(f[\langle s_1 \rangle, \ldots, \langle s_k \rangle])$ is uniquely determined. Moreover, using subtree-closedness, one can easily verify the following lemma by structural induction on t.

Lemma 2.1. *If Ω is a closed and consistent observation table, then $\delta(t) = \langle t \rangle$ for all $t \in \Sigma(S)$. In particular, for $t \in \Sigma(S)$, we have $t \in L(A_\Omega)$ if and only if $t \in L$.*

The learner $\mathrm{L}_*^{\mathrm{tfta}}$ starts with the observation table given by $S = \emptyset$ and $C = \{\square\}$. In its main loop, it first makes sure that Ω is closed and consistent. This is done by a straightforward procedure complete that adds appropriate trees and contexts to S and C, resp., until Ω is closed and consistent. Then, $\mathrm{L}_*^{\mathrm{tfta}}$ constructs A_Ω and passes it to the teacher in an equivalence query. If the teacher accepts it, learning has been successful. Otherwise, subtrees(t) is added to S and the next iteration starts. Whenever elements are added to S or C, the required membership queries are asked to fill the new cells (t, c) of the table with the membership information $\chi_L(c \cdot t)$.

Below follows the pseudo code of the learner. In this pseudo code, we denote an observation table by the components S and C:

```
procedure L*^tfta where Ω = (S, C)
  Ω := (∅, {□})
  loop
    complete(Ω);
    construct A_Ω;
    t := eqQuery(A_Ω);          (ask equivalence query)
    if t = ⊥ then return A_Ω
    else S := S ∪ subtrees(t)

procedure complete(S, C)
  loop
    if ∃c ∈ Σ_□(S), s, s' ∈ S: ⟨s⟩ = ⟨s'⟩ ∧ ⟨c · s⟩ ≠ ⟨c · s'⟩ then   (table inconsistent)
      choose d ∈ C with member(d · c · s) ≠ member(d · c · s');
      C := C ∪ {d · c}                                            (add witness to C)
    else if ∃t ∈ Σ(S) such that ⟨t⟩ ∉ ⟨S⟩ then                    (table not closed)
      S := S ∪ {t}
    else return;
```

Clearly, as long as Ω is not closed and consistent, each iteration of complete enlarges $\langle S \rangle$. In particular, complete terminates, because the index of L is finite. Now, consider the main procedure of $\mathrm{L}_*^{\mathrm{tfta}}$, and let Ω' be the new observation table Ω' obtained by adding subtrees(t) to S (where t is a counterexample). If Ω' would still be closed and consistent, on the one hand, it could easily be shown that $A_\Omega = A_{\Omega'}$. On the other hand, Lemma 2.1 would apply to $A_{\Omega'}$, stating that t is not a counterexample for $A_{\Omega'}$, contradicting the fact that it is a counterexample for A_Ω. Thus, Ω' cannot be closed and consistent. By the reasoning above, this means that the following call of complete enlarges $\langle S \rangle$. We conclude that $\mathrm{L}_*^{\mathrm{tfta}}$ terminates after at most n executions of the main loop, where n is the index of L.

Theorem 2.1 ([32]). *Let $A_L^{\mathrm{t}} = (\Sigma, Q, \delta, F)$. The learner $\mathrm{L}_*^{\mathrm{tfta}}$ returns an fta isomorphic to A_L^{t}, and runs in time polynomial in m^r and $|\delta|$, where m is the maximum size of counterexamples returned by the teacher, r is the maximum rank of symbols in Σ, and $|\delta|$ is the number of transitions.*

Note that the number $|Q|$ of states of A_L^{t} (i.e., the index n of L) does not occur in the preceding statement, because the totality of the fta implies that $|\delta| \geq |Q|$. Let us have a look at an example.

Example 2.1. Let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$, and consider the tree language L consisting of all trees in T_Σ that do not contain two nodes such that one is a child of the other and both are labelled with the same symbol.

The learner $\mathrm{L}_*^{\mathrm{tfta}}$ starts with the table $(\emptyset, \{\Box\})$, which is not closed, because $\langle S \rangle = \emptyset$ does not contain $\langle a \rangle$, but $a \in \Sigma(S)$. Thus, complete adds a to S. The resulting observation table is the first one shown in Figure 1. Here, the trees in S are those above the single horizontal line, and the trees in $\Sigma(S) \setminus S$ are those shown below it. The table is obviously closed and consistent, because all trees in $\Sigma(S)$ have equal rows. The transitions of the resulting automaton A_Ω are shown

to the left of the rows they result from. Since the state $\langle a \rangle$ is accepting (because $a \in L$, which is signified by the fact that $\langle a \rangle$ equals 1 at $\Box$), we have $L(A_\Omega) = \mathrm{T}_\Sigma$. Hence, the teacher may give the counterexample $t = g[g[a]]$. The table resulting from the addition of subtrees(t) to S is inconsistent, since the two trees shown in boldface letters have equal rows, whereas the trees they are subtrees of do not. After the addition of $g[\Box]$ to C, the table is closed and consistent. The resulting fta is passed to the teacher in another equivalence query, and the teacher returns a counterexample. Again, the table needs to be made consistent using complete. As the reader may check, the fta A_Ω obtained from the resulting table is isomorphic to A_L^{t}.

Let us say that a tree t is *live* (with respect to a recognizable tree language $L \subseteq \mathrm{T}_\Sigma$) if it occurs as a subtree of at least one tree in L. Otherwise, t is *dead*. As a direct consequence of this definition, the set of dead trees forms a congruence class of $\equiv_L$ (or is empty). The state of A_L^{t} corresponding to this congruence class is said to be the *dead state of* A_L^{t} (if it exists). The canonical *partial* fta recognizing L, denoted by A_L^{p}, is constructed in the same way as A_L^{t}, but taking as its state set the set $\{[t]_{\equiv_L} \mid t \in \mathrm{T}_\Sigma \text{ is live}\}$, and restricting the transition function accordingly. In other words, A_L^{p} is obtained from A_L^{t} by deleting its dead state, if it exists, and is equal to A_L^{t}, otherwise. If a computation of A_L^{t} reaches the dead state on one of the subtrees of the input tree, then this input tree cannot be accepted. Hence, we obviously have $L(A_L^{\mathrm{p}}) = L(A_L^{\mathrm{t}}) = L$. We shall now consider a learner that constructs A_L^{p} instead of A_L^{t}.

The learner $\mathrm{L}_*^{\mathrm{tfta}}$ has the advantage that it asks at most n equivalence queries, where n is the index of L. Its major disadvantages are that (a) S potentially contains a lot of redundant information, since all subtrees of all counterexamples received end up in S, and (b) the observation table contains $|\Sigma(S)|$ rows to make A_Ω total. Together, (a) and (b) are responsible for the appearance of m^r in Theorem 2.1. Moreover, A_L^{t} always contains at least n^r transitions, whereas the number of transitions of A_L^{p} may be much smaller. The learner $\mathrm{L}_*^{\mathrm{fta}}$ developed in [18] avoids these disadvantages at the price of potentially asking a considerably larger number of equivalence queries.

Even $\mathrm{L}_*^{\mathrm{fta}}$ uses an observation table. However, rather than indexing the rows by the trees in $\Sigma(S)$, they are now indexed by trees in a set T such that $S \subseteq T \subseteq \Sigma(S)$. Thus, this set T takes the role of $\Sigma(S)$, but will typically not contain all trees in $\Sigma(S)$. As before, columns are indexed by contexts from a finite set $C \subseteq \mathrm{C}_\Sigma$.

Since $S \subseteq T \subseteq \Sigma(S)$, both T and S are subtree-closed. In addition to this, $\mathrm{L}_*^{\mathrm{fta}}$ maintains the invariant that, for every tree $t \in T$, there is exactly one tree $s \in S$ such that $\langle s \rangle = \langle t \rangle$. This means that closedness and consistency do not need to be checked explicitly, because S never contains redundant information. As a consequence, $A_\Omega = (Q, \Sigma, \delta, F)$ can be defined as before, the only difference being that it is total only if it happens to be the case that $T = \Sigma(S)$. As the trees in S have pairwise distinct rows, the correspondences between S and Q and between T and δ (viewing δ as a set of transitions) are bijections. In particular, each transition is represented by a unique tree in T.

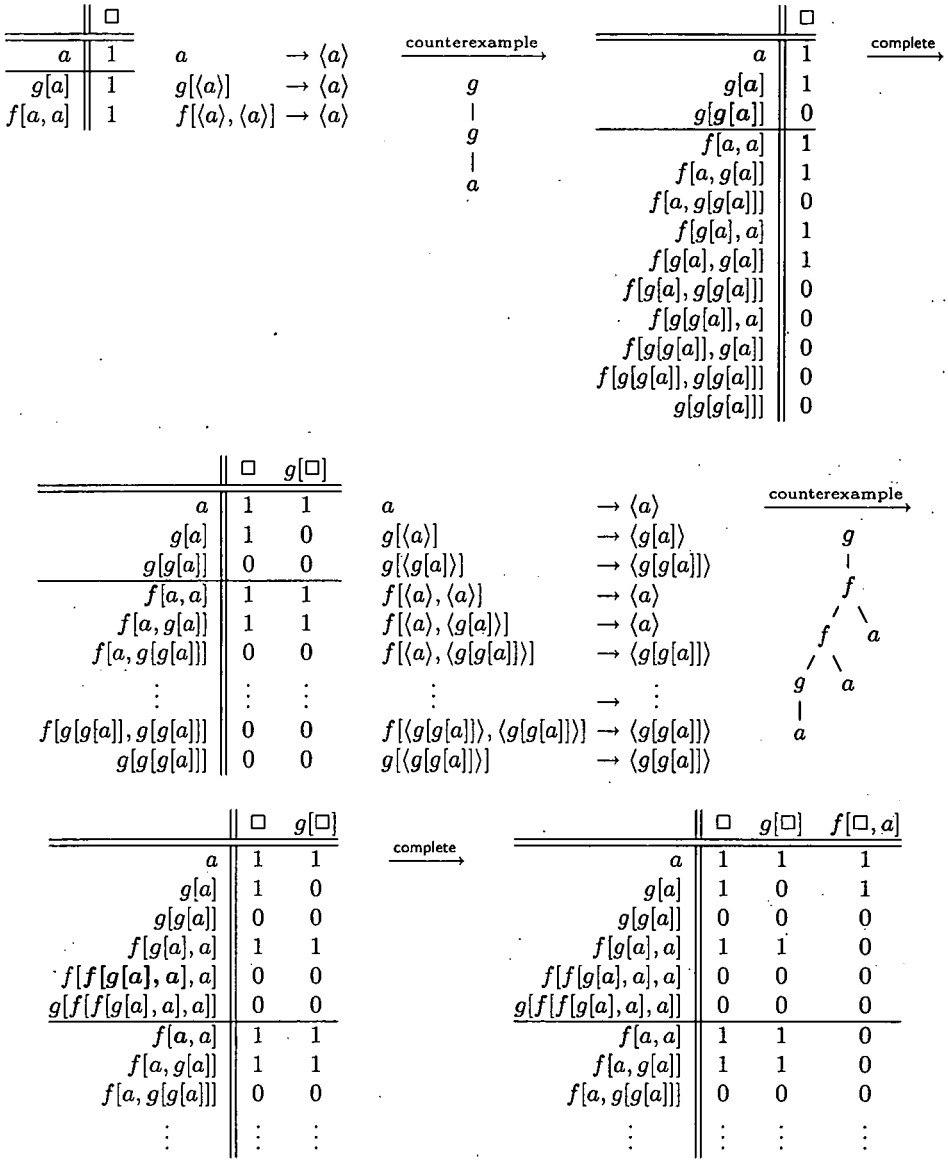


Figure 1: A run of L_*^{tfta}, showing (partial) observation tables, inconsistencies (in boldface letters), transitions resulting from the rows of consistent tables (except for the final table), and counterexamples that the teacher may choose to return.

Similar to L_*^{tfta}, L_*^{fta} starts with the observation table given by $S = \emptyset$ (and, thus, also $T = \emptyset$), and $C = \{\Box\}$. It repeatedly constructs A_Ω and asks an equivalence query. As long as a counterexample t is received, Ω is extended by a tree (and possibly a context) extracted from t, and the process continues:

```
procedure L*^fta where Ω = (S,T,C)
  Ω := (∅,∅,{□})
  loop
    construct A_Ω;
    t := eqQuery(A_Ω);          (ask equivalence query)
    if t = ⊥ then return A_Ω
    else Ω := extend(Ω,t)
```

The heart of L_*^{fta} is the procedure extend, which examines a counterexample in a bottom-up manner to find out where things go wrong, rather than adding all subtrees of t to S. The technique used for this was introduced by Shapiro [35] and is known as contradiction backtracking. The pseudo code looks like this:

```
procedure extend(Ω,t) where Ω = (S,T,C)
  loop
    decompose t into t = c·t' where t' = f[s_1,...,s_k] ∈ Σ(S) \ S;
    if t' ∈ T then
      let s be the unique tree in S with ⟨s⟩ = ⟨t'⟩
      if member(c·s) = member(t) then t := c·s          (case 1a)
      else return close(S,T,C ∪ {c})                   (case 1b)
    else return close(S,T ∪ {t'},C)                    (case 2)
```

Here, the decomposition of t into $c \cdot t'$ can be done by a simple algorithm that checks in a bottom-up manner which subtrees of t belong to S, and returns the first tree t' encountered which is not in S (but which, therefore, must necessarily be in $\Sigma(S)$). The procedure close is a simplified version of the procedure complete of L_*^{tfta}, corresponding to the second case in the latter. It checks the trees $t \in T$ one by one, and adds t to S if S does not yet contain a tree s with $\langle s \rangle = \langle t \rangle$. Let δ be the transition function of A_Ω. If $t' \in T$, then $\delta(c \cdot s) = \delta(c \cdot t') = \delta(t)$, because $\delta(s) = \langle s \rangle = \langle t' \rangle = \delta(t')$. In other words, A_Ω returns the same answer if run on t and $c \cdot s$. Together with the condition $\mathrm{member}(c \cdot s) = \mathrm{member}(t)$, this means that $c \cdot s$ is also a counterexample, in case 1a. In case 1b, we have found a context c that separates the trees s and t' that have been equivalent according to Ω. Finally, in case 2, we have found a missing transition.

The use of contradiction backtracking in extend makes sure that the trees in S represent pairwise distinct states, those in T represent pairwise distinct transitions, and the total number of contexts added does not exceed the number of states. Moreover, it guarantees that no dead tree is ever added to T. Indeed, only case 2 results in the addition of a tree t' to T. Since the transition represented by t' is not in T, we know that A_Ω rejects $t = c \cdot t'$. Hence, t must be a positive counterexample, which shows that t' is live.[3] These properties make L_*^{fta} quite efficient.

[3] This fact, showing that c is a so-called sign of life for t', will turn out to be of some importance in Section 3.

Theorem 2.2 ([18]). *The learner* $\mathrm{L}_*^{\mathrm{fta}}$ *returns an fta* (Σ, Q, δ, F) *isomorphic to* A_L^{p}*, and runs in time* $O(r \cdot |Q| \cdot |\delta| \cdot (|Q| + m))$*, where* m *is the maximum size of counterexamples returned by the teacher,* r *is the maximum rank of symbols in* Σ*, and* $|\delta|$ *is the number of transitions.*

The algorithm requires $|Q| + |\delta| + 1$ equivalence and $m + |Q| \cdot (|\delta| + 1)$ membership queries. As mentioned above, the number of equivalence queries asked is the major disadvantage of $\mathrm{L}_*^{\mathrm{fta}}$ in comparison with $\mathrm{L}_*^{\mathrm{tfta}}$. In practice, the number of equivalence queries used by $\mathrm{L}_*^{\mathrm{fta}}$ can often be reduced by re-using counterexamples [17]; see also the following example.

Example 2.2. Let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$ be as in Example 2.1, and consider the tree language L consisting of all trees of the form $c \cdot f[t, a]$, where $c \in \mathrm{C}_{\{g\}}$ and $t \in \mathrm{T}_{\{g,a\}}$. Thus, the trees in L consist of a chain of gs at the top, followed by a single f, whose first subtree is a chain of gs (ending in an a), whereas the second is a single a.

In the first step, the teacher will be given the empty automaton, which accepts the empty language. Suppose the teacher returns the left-most tree in Figure 2 as a counterexample. Searching for a subtree in $\Sigma(S) \setminus S$ in a bottom-up manner, we immediately encounter one of the leaves a and observe that it represents a missing transition (case 2). Therefore, a is added to T (and close adds it to S, because S does not yet contain any tree whose row is 0). Following $\mathrm{L}_*^{\mathrm{fta}}$ strictly, we would now build the new automaton A_Ω and ask the teacher a new equivalence query. However, since the current tree is still a counterexample (it is not accepted by the new automaton either), we can as well continue using the current tree (see [17]). We now find the subtree $g[a]$, which represents again a new transition, but not a new state. In the next iteration (again re-using the counterexample), we find that $g[a]$ is in T and can be replaced with a without invalidating the counterexample (case 1a). Thus, we continue with the third tree in Figure 2, and find that $f[a, a]$ represents a new transition and state. Finally, we also find that $g[f[a, a]]$ represents a transition. When this has happened, the automaton correctly accepts the tree, so that we have to ask a new equivalence query.

Suppose the teacher chooses the leftmost tree in the second row of Figure 2. We find that $g[a]$ cannot be replaced with a once more, because $f[a, a] \in L$ (case 1b). Consequently, $f[a, \Box]$ is a context that distinguishes between a and $g[a]$.

Finally, when processing the last counterexample, we first discover that $g[g[a]]$ represents a transition, and then that $f[g[a], a]$ represents another one. Now, an equivalence query reveals that the resulting automaton is the correct one.

Recently, Besombes and Marion [7] have proposed the learner $\mathrm{L}_*^{\mathrm{rep}}$ (called ALTEX in [7]), that avoids the use of equivalence queries. Instead, it exploits a set of positive examples in which all the transitions of the sought automaton are required to be represented (see also [2]). Intuitively, there is a close relation between the two learners, because $\mathrm{L}_*^{\mathrm{fta}}$ uses equivalence queries precisely in order to discover such representatives. It may be interesting to try to find out whether there is a deeper formal relationship.

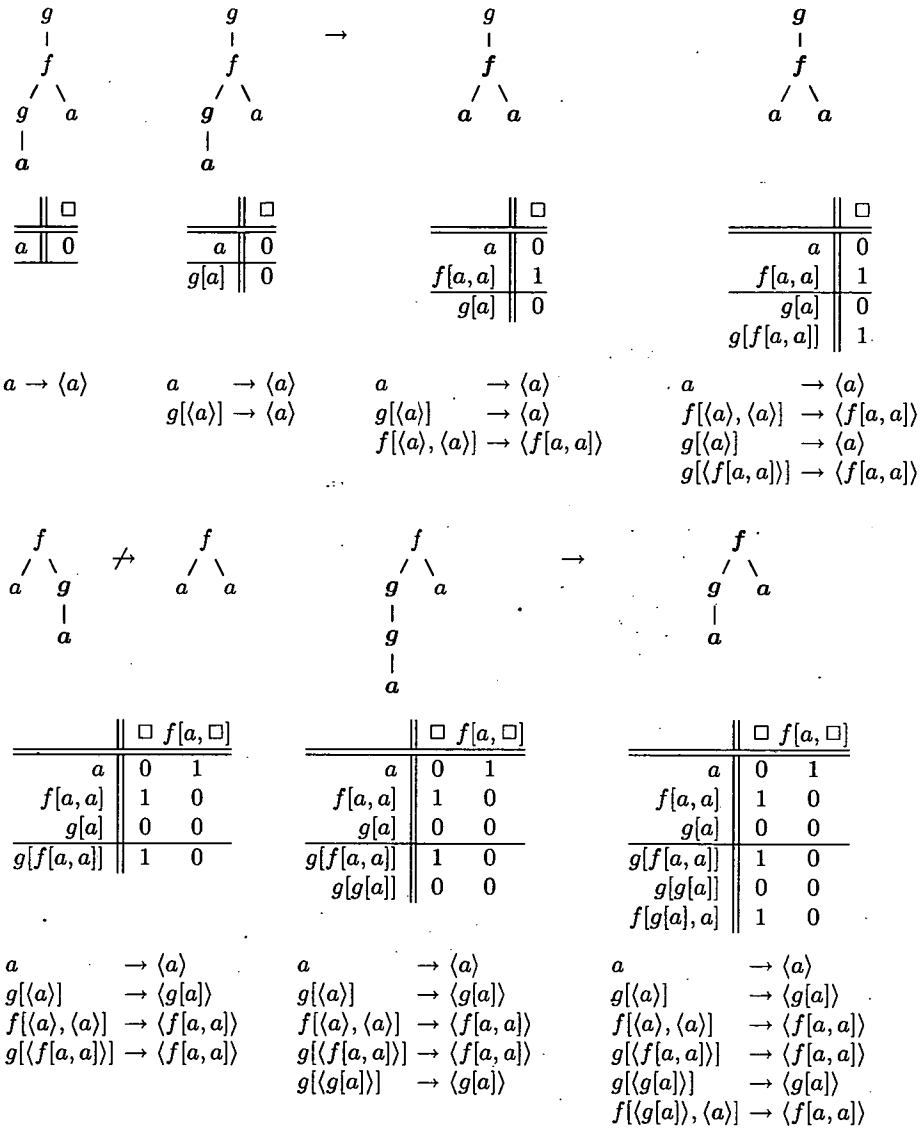


Figure 2: A run of L_*^{fta}, showing the trees inspected, the resulting observation tables, and the transitions. Steps according to case 1a (preserving the property of being a counterexample) are indicated by '$\to$', whereas $\not\to$ indicates steps according to case 1b (yielding a separating context).

Let us have a coarse look at $\mathrm{L}_*^{\mathrm{rep}}$. A set $R \subseteq \mathrm{T}_\Sigma$ is a *representative sample for* L if subtrees(R) contains, for every live tree $t = f[t_1, \ldots, t_k]$, a tree $t' = f[t'_1, \ldots, t'_k]$ such that $t'_1 \equiv_L t_1, \ldots, t'_k \equiv_L t_k$. In other words, the transition $f[[t_1]_{\equiv_L}, \ldots, [t_k]_{\equiv_L}] \to [t]_{\equiv_L}$ of the canonical fta is represented by a subtree of at least one of the trees in R. Now, learning starts with the observation table given by $T = \text{subtrees}(R)$ and $C = \{c \in \mathrm{C}_\Sigma \mid \exists t \in \mathrm{T}_\Sigma \colon c \cdot t \in R\}$. The set T is never going to change, and there is no distinguished subset S of trees representing states.

Somewhat similar to the situation in $\mathrm{L}_*^{\mathrm{tfta}}$, and in contrast to $\mathrm{L}_*^{\mathrm{fta}}$, Ω may be inconsistent, which now means that there are trees $t = f[t_1, \ldots, t_k]$ and $t' = f[t'_1, \ldots, t'_k]$ in T such that $\langle t_i \rangle = \langle t'_i \rangle$ for $i = 1, \ldots, k$, but $\langle t \rangle \neq \langle t' \rangle$. It can be shown that, in this case, there is an inconsistency with $t_i \equiv_L t'_i$ for all but one $i \in \{1, \ldots, k\}$. With this in mind, the situation becomes entirely similar to $\mathrm{L}_*^{\mathrm{tfta}}$: if j is the unique index with $t_j \not\equiv_L t'_j$, and $d \in C$ is a context separating t from t' (which exists because $\langle t \rangle \neq \langle t' \rangle$), then the context $d \cdot c$ with $c = f[t_1, \ldots, t_{j-1}, \Box, t_{j+1}, \ldots, t_k]$ separates t_j from t'_j.

The learner can now choose such a separating context d for every inconsistent pair of trees t and t' as above, and ask a membership query for each of the trees $d \cdot f[t_1, \ldots, t_{j-1}, t'_j, t_{j+1}, \ldots, t_k]$ ($j \in \{1, \ldots, k\}$), until the answer differs from the table entry for t in column d, to find c. In this way, a context $d \cdot c$ that separates t_j from t'_j is obtained.[4] Having found such a context, $\mathrm{L}_*^{\mathrm{rep}}$ adds it to C and checks again whether the observation table is consistent. Since the index of L is finite, the process must eventually terminate, yielding a consistent table. This table gives rise to an fta A_Ω in a similar manner as before. For a consistent table, using the fact that every transition is represented in $T = \text{subtrees}(R)$, it can be shown by induction on the size of minimal separating contexts that, for $t, t' \in T$, if $\langle t \rangle = \langle t' \rangle$, then $t \equiv_L t'$. From this, it follows easily that A_Ω is isomorphic to A_L^{p}.[5]

Theorem 2.3 ([7]). *The learner* $\mathrm{L}_*^{\mathrm{rep}}$ *returns an fta* (Σ, Q, δ, F) *isomorphic to* A_L^{p} *in time polynomial in* $\sum_{t \in R} |t|$ *(where* $|t|$ *denotes the size of* t*).*

Let us have a look at an example.

Example 2.3. Let $\Sigma = \{f^{(2)}, a^{(0)}, b^{(0)}\}$ and $L = \mathrm{T}_\Sigma \setminus (\mathrm{T}_{\{f,a\}} \cup \{b\})$, i.e., L contains all trees over Σ of size greater than one that contain at least one b. The canonical fta contains states q_a, q_b, q_{f}, where q_{f} is final. Its transition table is

$$\delta(t) = \begin{cases} q_a & \text{if } t \in \{a, f[q_a, q_a]\} \\ q_b & \text{if } t = b \\ q_{\mathrm{f}} & \text{otherwise.} \end{cases}$$

The set R of trees shown in Figure 3 is a representative sample. Building the

[4] Alternatively, following the description in [7], the learner could simply pick any inconsistent pair t, t' as above and a separating context d, and add all contexts $d \cdot f[t_1, \ldots, t_{j-1}, \Box, t_{j+1}, \ldots, t_k]$ to C, because it will eventually also encounter the right one and include it. However, it seems clear that this may have a negative impact on the efficiency.

[5] The proof of this fact given in [7, Lemma 5] does not seem to be convincing, but it is easily corrected by the inductive argument mentioned, showing that $\langle t \rangle = \langle t' \rangle$ implies $t \equiv_L t'$.

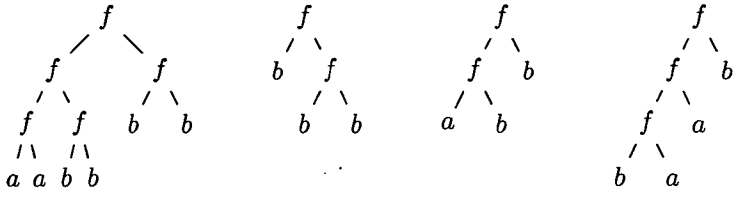


Figure 3: A representative sample R

corresponding initial observation table, we see that $\sim_\Omega$ divides $T = \text{subtrees}(R)$ into two equivalence classes, namely $T \setminus L = \{a, b, f[a,a]\}$ and $T \cap L$. The reason is that, among the contexts in C, only $\square$ separates any trees at all, because every $c \in C \setminus \{\square\}$ (i.e., every context obtained from a tree in R by replacing a proper subtree with $\square$) contains a b, which means that $\chi_L(c \cdot t) = 1$ for all $t \in \mathrm{T}_\Sigma$.

Thus, we should be able to find a pair of trees in T revealing an inconsistency. Indeed, there are three, obtained by combining $f[a,b], f[b,a], f[b,b] \in L$ with $f[a,a] \notin L$. This gives rise to the context $f[a,\square]$ separating a from b. Of course, $f[\square,a]$ would do as well, but it may be interesting to note that neither $f[\square,b]$ nor $f[b,\square]$ does (see also footnote 4). As the reader may wish to verify, the table Ω' enlarged by this context is consistent, and $A_{\Omega'}$ is isomorphic to A_L^{p}.

3 Learning Tree Series

It is now a natural step to wonder whether learning of recognizable tree series is possible as well. The number of papers addressing this problem is still rather small. One may roughly divide them into two categories. The first deals with the special case of stochastic tree automata, weighted tree automata (wta) with weights in $[0,1]$ that compute a probability distribution on T_Σ. This case is of particular interest because stochastic languages play an important role in, e.g., natural language processing. To learn stochastic tree languages, it is probably most natural to consider a learning-from-text-like setting: positive examples are drawn according to a probability distribution D, and the goal is to learn D in the limit by, e.g., constructing an appropriate wta. A learner of this kind has recently been presented by Denis and Habrard [16].

The second category of learners does not assume that the sought wta is a stochastic tree automaton. There seem to be only two results of this kind, both using the MAT model and the general algorithmic idea explained in the previous section. Let us first give some basic definitions. Readers who wish to read a more decent introduction to weighted tree automata are referred to the excellent survey by Fülöp and Vogler [22].

Let $\mathbf{S} = (\mathbb{S}, +, \cdot, 0, 1)$ be a (commutative) semiring, i.e., a set $\mathbb{S}$ together with binary addition and multiplication operations $+$ and $\cdot$ and distinct elements $0, 1 \in \mathbb{S}$ such that $(\mathbb{S}, +, 0)$ and $(\mathbb{S}, \cdot, 1)$ are commutative monoids, multiplication distributes over addition, and 0 is absorbing with respect to multiplication. From now on, we

simply denote $\mathbf{S}$ by $\mathbb{S}$. A *tree series* is a mapping $\psi\colon \mathrm{T}_\Sigma \to \mathbb{S}$. Given such a tree series, we call the set $\mathrm{supp}(\psi) = \{t \in \mathrm{T}_\Sigma \mid \psi(t) \neq 0\}$ the *support of* ψ.

Below, for a finite index set I, we let $\mathbb{S}^I$ denote the set of all vectors over $\mathbb{S}$ indexed by I. As usual, the ith component of $v \in \mathbb{S}^I$ is denoted by v_i, for $i \in I$. The inner product of $u, v \in \mathbb{S}^I$ is $u \cdot v = \sum_{i \in I} u_i \cdot v_i$.

Definition 3.1. *Let $\mathbb{S}$ be a semiring. A* weighted tree automaton *(wta) over $\mathbb{S}$ is a tuple $A = (\Sigma, Q, \mu, \lambda)$ consisting of a ranked alphabet Σ, a ranked alphabet Q of states such that $Q = Q_{(0)}$, a transition weight table $\mu \in \mathbb{S}^{\Sigma(Q) \times Q}$, and a root weight mapping $\lambda \in \mathbb{S}^Q$. Thus, μ assigns a weight μ_τ to every transition $\tau \in \Sigma(Q) \times Q$. A is (bottom-up)* deterministic *(a dwta) if, for every $l \in \Sigma(Q)$, there is at most one $q \in Q$ such that $\mu_{l \to q} \neq 0$.*

For $t = f[t_1, \ldots, t_k] \in \mathrm{T}_\Sigma$, we define $\widehat{\mu}(t) \in \mathbb{S}^Q$ by setting

$$\widehat{\mu}(t)_q = \sum_{q_1, \ldots, q_k \in Q} \mu_{f[q_1, \ldots, q_k] \to q} \cdot \prod_{i=1,\ldots,k} \widehat{\mu}(t_i)_{q_i},$$

for all $q \in Q$.

The tree series recognized by A is given by $\psi_A(t) = \lambda \cdot \widehat{\mu}(t)$, and is called a recognizable tree series.

In the following, we want to consider the problem of learning a wta in the MAT model, first for dwta over a semifield, and then for nondeterministic wta over a field. Clearly, for this to be possible, the teacher has to be given appropriate capabilities. Thus, if $\mathcal{A}$ is the class of wta to be learned, and ψ is the target series, membership queries become *coefficient queries*: given a tree $t \in T_\Sigma$, the procedure $\mathsf{coef}(t)$ will return $\psi(t)$. Similarly, equivalence queries have to be extended: the input is a wta $A \in \mathcal{A}$, and $\mathsf{eqQuery}(A)$ will either return $\perp$, indicating that $\psi_A = \psi$, or a counterexample, a tree $t \in \mathrm{T}_\Sigma$ such that $\psi_A(t) \neq \psi(t)$.

As mentioned, we are first going to have a look at the deterministic case. For readers who are not yet familiar with wta, a small example (which will be continued later) follows.

Example 3.1. We consider the semifield $\mathbb{S} = (\mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0)$. To avoid confusion, the reader should keep in mind that $+$ plays the role of multiplication in this example, with ∞ being the absorbing element, and 0 being the neutral element.

As in Example 2.2, let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$. For a tree t of the form $c \cdot f[t', a]$, where $c \in \mathrm{C}_{\{g\}}$ and $t' \in \mathrm{T}_{\{g,a\}}$, let $\psi(t) = 2m + n$, where m is the number of occurrences of g in c, and n is the size of t'. For all other trees $t \in \mathrm{T}_\Sigma$, let $\psi(t) = \infty$. Thus, the support of ψ is the tree language in Example 2.2.

A dwta over $\mathbb{S}$ recognizing ψ can be constructed by using states q_1, q_2, q_3. Except for the addition of weights, the automaton is the same as the one in Example 2.2. It will be in state q_1 when it has just read an a, in state q_2 when it has read a number of gs above an a, and in state q_3 when it has read a tree in $\mathrm{supp}(\psi)$. For the specification of concrete dwta, it is convenient to write μ as a set of rules of the form $l \xrightarrow{w} q$, where $l \in \Sigma(Q)$ and q is the unique element of Q such that $w = \mu_{l \to q}$

is non-zero (which, in the present case, means that $w \neq \infty$). Using this notation, A contains the following rules:

$$a \xrightarrow{0} q_1, \qquad g[q_1] \xrightarrow{1} q_2, \qquad g[q_2] \xrightarrow{1} q_2,$$
$$f[q_1, q_1] \xrightarrow{1} q_3, \quad f[q_2, q_1] \xrightarrow{1} q_3, \quad g[q_3] \xrightarrow{2} q_3.$$

Furthermore, $\lambda_{q_1} = \lambda_{q_2} = \infty$ and $\lambda_{q_3} = 0$.

Now, let us have a look at the MAT learner $\mathrm{L}_*^{\mathrm{dwta}}$ for dwta over a (commutative) semifield $\mathbb{S}$ by Maletti [30]. It extends $\mathrm{L}_*^{\mathrm{fta}}$ to the weighted case and generalizes an earlier version proposed by Drewes and Vogler [19], which was restricted to the class of "all-accepting" dwta.

The learner $\mathrm{L}_*^{\mathrm{dwta}}$ makes use of the Myhill-Nerode theorem for deterministically recognizable tree series over commutative semifields [8]. Thus, from now on, every $a \in \mathbb{S} \setminus \{0\}$ is assumed to have a multiplicative inverse. As in $\mathrm{L}_*^{\mathrm{fta}}$, observation tables are given by sets $S, T \subseteq \mathrm{T}_\Sigma$ and $C \subseteq \mathrm{C}_\Sigma$. The entry in row t and column c is now the coefficient $\psi(c \cdot t)$. The fact that $\mathrm{L}_*^{\mathrm{fta}}$, in T, only collects live trees becomes now crucial for the correctness of the learner. In the context of tree series, a tree $t \in \mathrm{T}_\Sigma$ is live if there exists a *sign of life for* t, a context $c \in \mathrm{C}_\Sigma$ such that $\psi(c \cdot t) \neq 0$. The case of tree series poses a difficulty not present in the language case: if $\widehat{\mu}(t)_q \neq 0$ but $\lambda_q = 0$, then the value of $\widehat{\mu}(t)_q$ is hidden in the sense that a coefficient query on t will yield $\psi(t) = 0$. To determine the right coefficients during the construction of A_Ω, we thus have to make sure that C contains a sign of life, for every $t \in T$. In the algorithm extend, this is easily guaranteed by changing case 2 in such a way that c is added to C (see footnote 3 on p. 258).

Thus, the crucial invariant maintained by $\mathrm{L}_*^{\mathrm{dwta}}$ is that, as in $\mathrm{L}_*^{\mathrm{fta}}$, the observation table $\Omega = (S, T, C)$ satisfies $S \subseteq T \subseteq \Sigma(S)$. In addition, C now contains a sign of life for every tree in T. For $t, t' \in T$, what used to be the equality of $\langle t \rangle$ and $\langle t' \rangle$ in the unweighted setting, is now replaced by the requirement that one row be a multiple of the other. More precisely, let $\langle t \rangle \approx \langle t' \rangle$ if and only if there exists an $a \in \mathbb{S}$ such that $\langle t \rangle = a \cdot \langle t' \rangle$ (where $a \cdot \langle t' \rangle$ denotes the scalar multiplication of the row $\langle t' \rangle$ by a). Note that, due to the existence of signs of life, a is non-zero and is uniquely determined for every pair of trees in T (if it exists). Similar to $\mathrm{L}_*^{\mathrm{fta}}$, for every tree $t \in T$, S will always contain exactly one tree s such that $\langle s \rangle \approx \langle t \rangle$. Given $t \in T$, we will denote this particular tree $s \in S$ by $\mathrm{rep}_\Omega(t)$.

Now, we can assign a weight $\psi_\Omega(t)$ to every tree $t \in T$: $\psi_\Omega(t)$ is the unique factor $a \in \mathbb{S}$ such that $\langle t \rangle = a \cdot \langle \mathrm{rep}_\Omega(t) \rangle$. In particular, $\psi_\Omega(s) = 1$ for every $s \in S$.[6]

Using these definitions, an observation table $\Omega = (S, T, C)$ gives rise to the dwta $A_\Omega = (\Sigma, Q, \mu, \lambda)$, where

- $Q = \langle S \rangle$,
- for every transition $\tau = (f[\langle s_1 \rangle, \ldots, \langle s_k \rangle] \to \langle t \rangle)$, where $t = f[s_1, \ldots, s_k] \in T$, we let $\mu_\tau = \psi_\Omega(t)$,

[6] This definition of $\psi_\Omega(t)$ differs from the one given in [30], but fulfills the same purpose. This illustrates the fact that there may be several minimal wta recognizing ψ, which differ in their transition weights (and in λ).

- all remaining transition weights μ_τ are 0, and
- $\lambda_{\langle s \rangle} = \psi(s)$ for all $s \in S$.

In the same way as $\mathrm{L}_*^{\mathrm{fta}}$, $\mathrm{L}_*^{\mathrm{dwta}}$ now starts with the observation table $\Omega = (\emptyset, \emptyset, \{\Box\})$. It repeatedly constructs A_Ω, asks an equivalence query, and passes the counterexample received (if any) to the procedure extend. In other words, the main procedure of $\mathrm{L}_*^{\mathrm{dwta}}$ looks exactly like that of $\mathrm{L}_*^{\mathrm{fta}}$ (although it does now work with dwta rather than fta, of course). Even extend looks quite the same as before, the major difference being that we now add the context c as a sign of life in case 2:

```
procedure extend(Ω, t) where Ω = (S, T, C)
  loop
    decompose t into t = c · t' where t' = f[s_1, ..., s_k] ∈ Σ(S) \ S;
    let s = rep_Ω(t');
    if t' ∈ T then
      if coef(t) = ψ_Ω(t') · coef(c · s) then t := c · s          (case 1a)
      else return close(S, T, C ∪ {c})                            (case 1b)
    else return close(S, T ∪ {t'}, C ∪ {c})                       (case 2)
```

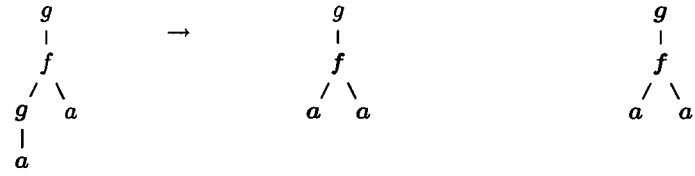
The following result, similar to Theorem 2.2, holds under the assumption that all relevant operations on $\mathbb{S}$ (addition, multiplication, and taking inverses) can be computed in constant time. Compared to Theorem 2.2, an additional factor $|Q|$ results from the fact that rows are not bit strings anymore, and thus cannot be stored as single integers.

Theorem 3.1 ([30]). *The learner $\mathrm{L}_*^{\mathrm{dwta}}$ returns a minimal dwta $A = (\Sigma, Q, \mu, \lambda)$ recognizing ψ in time $O(r \cdot |Q|^2 \cdot |\delta| \cdot (|Q| + m))$, where m is the maximum size of counterexamples returned by the teacher, r is the maximum rank of symbols in Σ, and $|\delta|$ is the number of transitions $\tau \in \Sigma(Q) \times Q$ such that $\mu_\tau \neq 0$.*

Let us have a look at an example.

Example 3.2. Consider the tree series ψ in Example 3.1, where, again, $\mathbb{S} = (\mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0)$. We now apply $\mathrm{L}_*^{\mathrm{dwta}}$ in order to construct, by means of learning, a dwta over $\mathbb{S}$ recognizing ψ. The counterexamples used as well as the states and transitions discovered are the same as in Example 2.2. In particular, counterexamples are re-used if possible. Furthermore, the context c in case 2 of extend is not added to the table if the table already contains a sign of life for t'. To save space in Figure 4, the very first step, in which a is found to be a new state and transition, is omitted. Otherwise, the figure is very similar to Figure 2. Indeed, the resulting wta recognizes ψ, as the reader may easily check, although the transition weights differ from those used in Example 3.1.

It seems to be clear that the learner $\mathrm{L}_*^{\mathrm{rep}}$ discussed in the previous section carries over to deterministic wta over $\mathbb{S}$ in quite exactly the same way as $\mathrm{L}_*^{\mathrm{fta}}$. Thus, the resulting learner would use coefficient queries and a representative sample, the latter being a subset of $\mathrm{supp}(\psi)$ covering every transition of a minimal dwta recognizing



	$\square$	$g[f[g[\square],a]]$
a	∞	4
$g[a]$	∞	5

$a \xrightarrow{0} \langle a\rangle$
$g[\langle a\rangle] \xrightarrow{1} \langle a\rangle$
$\lambda_{\langle a\rangle} = \infty$

	$\square$	$g[f[g[\square],a]]$
a	∞	4
$f[a,a]$	1	∞
$g[a]$	∞	5

$a \xrightarrow{0} \langle a\rangle$
$f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle$
$g[\langle a\rangle] \xrightarrow{1} \langle a\rangle$
$\lambda_{\langle a\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$

	$\square$	$g[f[g[\square],a]]$
a	∞	4
$f[a,a]$	1	∞
$g[a]$	∞	5
$g[f[a,a]]$	3	∞

$a \xrightarrow{0} \langle a\rangle$
$f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle$
$g[\langle a\rangle] \xrightarrow{1} \langle a\rangle$
$g[\langle f[a,a]\rangle] \xrightarrow{2} \langle f[a,a]\rangle$
$\lambda_{\langle a\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$

	$\square$	$g[f[g[\square],a]]$	$f[a,\square]$
a	∞	4	1
$f[a,a]$	1	∞	∞
$g[a]$	∞	5	∞
$g[f[a,a]]$	3	∞	∞

$a \xrightarrow{0} \langle a\rangle$
$f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle$
$g[\langle a\rangle] \xrightarrow{0} \langle g[a]\rangle$
$g[\langle f[a,a]\rangle] \xrightarrow{2} \langle f[a,a]\rangle$
$\lambda_{\langle a\rangle} = \lambda_{\langle g[a]\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$

	$\square$	$g[f[g[\square],a]]$	$f[a,\square]$
a	∞	4	1
$f[a,a]$	1	∞	∞
$g[a]$	∞	5	∞
$g[f[a,a]]$	3	∞	∞
$g[g[a]]$	∞	6	∞
$f[g[a],a]]$	2	∞	∞

$a \xrightarrow{0} \langle a\rangle$
$g[\langle a\rangle] \xrightarrow{0} \langle g[a]\rangle$
$g[\langle f[a,a]\rangle] \xrightarrow{2} \langle f[a,a]\rangle$
$g[\langle g[a]\rangle] \xrightarrow{1} \langle g[a]\rangle$
$f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle$
$f[\langle g[a]\rangle,\langle a\rangle] \xrightarrow{1} \langle f[a,a]\rangle$
$\lambda_{\langle a\rangle} = \lambda_{\langle g[a]\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$

Figure 4: A run of $\mathrm{L}_*^{\mathrm{dwta}}$, similar to the run of $\mathrm{L}_*^{\mathrm{fta}}$ in Figure 2.

ψ. Note that, even though there may be various minimal dwta recognizing ψ, their representative sets coincide, because two minimal dwta recognizing the same tree series over $\mathbb{S}$ differ only in the weights of their transitions (and in their root weights).

We now turn to the second learner for recognizable tree series, proposed by Habrard and Oncina [26]. In contrast to the one explained above (and, in fact, also in contrast to all other extensions of L_* known to the author), this learner works for nondeterministic wta. This becomes possible by making the stronger assumption that $\mathbb{S}$, the semiring considered, is a field. Thus, from now on, $\mathbb{S}$ is even assumed to have additive inverses. From the point of view of MAT learning, the important consequence of this assumption is that we, again, can make use of a Myhill-Nerode theorem; see [22, Theorem 3.31].

Below, since we are now dealing with nondeterministic wta $A = (\Sigma, Q, \mu, \lambda)$, it is occasionally convenient to specify μ as a function $\mu\colon \Sigma(Q) \to \mathbb{S}^Q$. The connection between the two views is, of course, that $\mu_{l\to q} = \mu(l)_q$ for all $l \in \Sigma(Q)$ and $q \in Q$.

Before turning to the discussion of the learner, let us have a look at an example of a nondeterministic wta.

Example 3.3. Let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$ and $\mathbb{S} = \mathbb{Q}$, where addition and multiplication are as usual. For a tree $t \in T_\Sigma$, let $\psi(t) = m + n$, where n is the number of nodes labelled f in t, and m is the number of nodes labelled f in t that do not have a child node labelled f. In other words, we count fs, and those which do not have another f as a direct descendant are counted twice. A minimal wta $A = (\Sigma, Q, \mu, \lambda)$ recognizing ψ has three states q_1, q_2, q_3. The intuition behind them is as follows. At the root of a (sub-)tree t, state q_1 carries the weight $w = 0$ if the root of t is labelled f, and $w = 1$ otherwise. At the same time, q_2 carries the weight $1 - w$. State q_3 always carries the weight $\psi(t)$. Consequently, denoting $v \in \mathbb{S}^Q$ as $(v_{q_1}, v_{q_2}, v_{q_3})$, the specification of μ reads as follows:

$$
\begin{array}{lll}
\mu(a) = (1,0,0) & & \\
\mu(g[q_1]) = (1,0,0) & \mu(f[q_1,q_1]) = (0,1,2) & \\
\mu(g[q_2]) = (1,0,0) & \mu(f[q,q']) = (0,1,1) & \text{if } q_2 \in \{q,q'\} \subseteq \{q_1,q_2\} \\
\mu(g[q_3]) = (0,0,1) & \mu(f[q,q']) = (0,0,1) & \text{if } \{q,q'\} \in \{\{q_1,q_3\},\{q_2,q_3\}\}.
\end{array}
$$

The root weights are given by $\lambda = (0,0,1)$. Figure 5 illustrates a computation.

Now, suppose that ψ is a recognizable tree series over a field $\mathbb{S}$. For the moment, let Ω denote the infinite observation table obtained by taking all of T_Σ as T (indexing the rows) and all of C_Σ as C (indexing the columns). Then the rank of Ω, viewed as a matrix, is finite. Moreover, it is not difficult to show that, for every set $S \subseteq T_\Sigma$, if there exists a tree $t \in T_\Sigma$ such that $\langle t \rangle$ is linearly independent of $\langle S \rangle$, then a tree with this property can even be found in $\Sigma(S)$. Therefore, there is a finite subtree-closed set[7] $S \subseteq T_\Sigma$ such that, for all $s \in S$, $\langle s \rangle$ is linearly independent of $\langle S \rangle \setminus \{\langle s \rangle\}$, and every row in $\langle T_\Sigma \rangle$ is a linear combination of rows in $\langle S \rangle$.

[7] Recall that subtree-closedness of S means that S even contains all subtrees of trees in S.

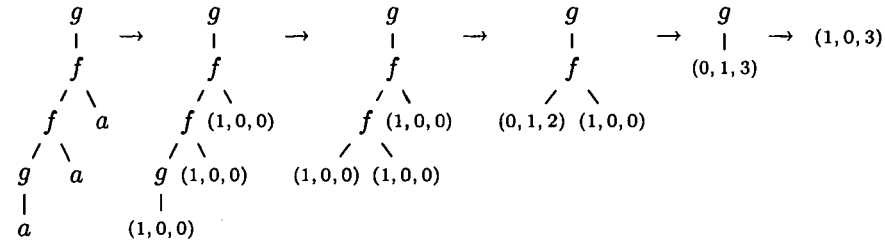


Figure 5: A computation of the wta in Example 3.3.

Assume that we have discovered such a set S, and let $Q = \langle S \rangle$. For every tree $t \in \mathrm{T}_\Sigma$, let $\widetilde{\mu}(t) \in \mathbb{S}^Q$ be the unique vector such that $\langle t \rangle = \sum_{s \in S} \widetilde{\mu}(t)_{\langle s \rangle} \cdot \langle s \rangle$. In other words, $\widetilde{\mu}(t)$ is the vector of coefficients of $\langle t \rangle$, if expressed as a linear combination of rows in $\langle S \rangle$. Then, for a tree t and a context c, we have

$$\psi(c \cdot t) = \Omega(t, c) = \sum_{s \in S} \widetilde{\mu}(t)_{\langle s \rangle} \cdot \Omega(s, c) = \sum_{s \in S} \widetilde{\mu}(t)_{\langle s \rangle} \cdot \psi(c \cdot s).$$

In particular, choosing $c = \Box$ and setting $\lambda_{\langle s \rangle} = \psi(s)$, we get $\psi(t) = \lambda \cdot \widetilde{\mu}(t)$. Since this is just the definition of $\psi_A(t)$, it remains to show how to discover S, together with a weight table μ such that $\widehat{\mu} = \widetilde{\mu}$.

This is done as follows, again using an observation table. As in $\mathrm{L}_*^{\mathrm{tfta}}$, rows are indexed by the trees in $\Sigma(S)$, i.e., $\Sigma(S)$ plays the role of T. Each time new contexts have been added to C, the learner makes sure that the table is closed, which now means that $\langle t \rangle$ is a linear combination of $\langle S \rangle$, for every tree $t \in \Sigma(S)$. Closedness can be achieved by an straightforward iterative procedure close that preserves subtree-closedness. Given that Ω is closed, a corresponding wta $A_\Omega = (\Sigma, Q, \mu, \lambda)$ with $Q = \langle S \rangle$ can be obtained along the lines of the preceding discussion: for every tree $l = f[\langle s_1 \rangle, \ldots, \langle s_k \rangle] \in \Sigma(Q)$, we let $\mu(l)$ be the unique vector such that $\langle t \rangle = \sum_{s \in S} \mu(l)_{\langle s \rangle} \cdot \langle s \rangle$. Furthermore, $\lambda_{\langle s \rangle} = \psi(s)$ for all $s \in S$.

Now, here is the pseudo-code of the main routine of the learner:

```
procedure L*^wta
  Ω = (S,C) := (∅,∅)
  loop
    construct A_Ω;
    t := eqQuery(A_Ω);      (ask equivalence query)
    if t = ⊥ then return A_Ω;
    else
      C := C ∪ {c ∈ C_Σ | ∃t' ∈ T_Σ: c·t' = t};
      S := close(S);
```

Thus, when a counterexample is received, C is enlarged by all contexts obtained from this counterexample. Since it can be shown that this increases the rank of Ω, termination is guaranteed. (In fact, the learner in [26] is slightly more optimized than the version described here. Before asking a new equivalence query, it is checked

whether there is a tree $t \in T$ such that $\psi_{A_\Omega}(t) \neq \psi(t)$. In other words, there is a context $c \in C$ such that $c \cdot t$ is a counterexample. In this case, the learner can obviously proceed by using $c \cdot t$ as a counterexample, thus avoiding the need to ask an equivalence query.)

Every counterexample increases $|S|$, which never gets larger than the number of states of a minimal wta recognizing ψ. Furthermore, every counterexample t leads to the inclusion of at most $|t|$ new contexts in C. As all the basic steps in the algorithm can be performed in polynomial time in the size of Ω (i.e, in $|\Sigma(S)|+|C|$), we get the following theorem.

Theorem 3.2 ([26]). *For every recognizable tree series over* $\mathbb{S}$, $\mathrm{L}_*^{\mathrm{wta}}$ *learns a minimal wta* A *recognizing* ψ *in polynomial time with respect to the size of* A *and the size of the largest counterexample returned by the teacher.*

Again, let us have a look at an example.

Example 3.4. We apply $\mathrm{L}_*^{\mathrm{wta}}$ to the tree series in Example 3.3. The initial wta, without any states, assigns the weight 0 to all trees. The teacher may respond with the counterexample $f[a,a]$, which leads to the first observation table in Figure 6. In the figure, only as many contexts of C are shown as needed. For example, $f[a,\Box]$ is left out in the first table. Furthermore, for the sake of clarity, the part below the horizontal line in each table lists all of T, rather than only $T \setminus S$.

The teacher may now give the counterexample $t = f[f[f[a,a],a],f[a,a]]$, because $\psi_{A_\Omega}(t) = 27/4$ rather than 6. Of the contexts obtained from t, we need only $f[f[f[\Box,a],a],f[a,a]]$ to distinguish between three states; see the second table in Figure 6. A_Ω now recognizes ψ, even though the "intuition" of the learner differs from the one used to construct the (equivalent) wta in Example 3.3. More precisely, let $\mathrm{root}_f(t)$ be the predicate which is true if and only if the root symbol of t is f. Then, if $\mu(t) = (v_1, v_2, v_3)$, we have

$$v_1 = 1 - \psi(t)/2, \qquad v_2 = \begin{cases} 1 & \text{if } \mathrm{root}_f(t) \\ 0 & \text{otherwise,} \end{cases} \qquad v_3 = \begin{cases} \psi(t)/2 - 1 & \text{if } \mathrm{root}_f(t) \\ \psi(t)/2 & \text{otherwise.} \end{cases}$$

Indeed, given the choice of λ, this means that A_Ω recognizes ψ.

4 Final Remarks

We have considered a family of grammatical inference algorithms for tree languages and tree series that can be regarded as more or less direct descendants of the learner L_* proposed by Angluin in [3]. An approach that has not been discussed here is the one presented in [6, 38] for string and tree languages, respectively (see also [37, 36]). This approach uses so-called correction queries instead of membership queries. Given a recognizable tree language $L \subseteq \mathrm{T}_\Sigma$ to be learned, a correction query $\mathsf{correct}(t)$ (where $t \in \mathrm{T}_\Sigma$) is answered by returning the smallest context $c \in \mathrm{C}_\Sigma$ such that $c \cdot t \in L$. Here, contexts are ordered according to a Knuth-Bendix order. A special token is returned if no c with $c \cdot t \in L$ exists, i.e., in case t is

	$\square$	$f[\square, a]$	
$s_1 = a$	0	2	$\lambda_{\langle s_1 \rangle} = 0$
$s_2 = f[a,a]$	2	3	$\lambda_{\langle s_2 \rangle} = 2$
a	0	2	$\mu(a) = (1,0)$
$g[s_1]$	0	2	$\mu(g[\langle s_1 \rangle]) = (1,0)$
$g[s_2]$	2	4	$\mu(g[\langle s_2 \rangle]) = (\frac{1}{2},1)$
$f[s_1,s_1]$	2	3	$\mu(f[\langle s_1 \rangle, \langle s_1 \rangle]) = (0,1)$
$f[s_1,s_2]$	3	4	$\mu(f[\langle s_1 \rangle, \langle s_2 \rangle]) = (-\frac{1}{4},\frac{3}{2})$
$f[s_2,s_1]$	3	4	$\mu(f[\langle s_2 \rangle, \langle s_1 \rangle]) = (-\frac{1}{4},\frac{3}{2})$
$f[s_2,s_2]$	5	6	$\mu(f[\langle s_2 \rangle, \langle s_2 \rangle]) = (-\frac{3}{4},\frac{5}{2})$

	$\square$	$f[\square,a]$	$f[f[f[\square,a],a],f[a,a]]$	
$s_1 = a$	0	2	6	$\lambda_{\langle s_1 \rangle} = 0$
$s_2 = f[a,a]$	2	3	7	$\lambda_{\langle s_2 \rangle} = 2$
$s_3 = g[f[a,a]]$	2	4	8	$\lambda_{\langle s_3 \rangle} = 2$
a	0	2	6	$\mu(a) = (1,0,0)$
$g[s_1]$	0	2	6	$\mu(g[\langle s_1 \rangle]) = (1,0,0)$
$g[s_2]$	2	4	8	$\mu(g[\langle s_2 \rangle]) = (0,0,1)$
$g[s_3]$	2	4	8	$\mu(g[\langle s_3 \rangle]) = (0,0,1)$
$f[s_1,s_1]$	2	3	7	$\mu(f[\langle s_1 \rangle, \langle s_1 \rangle]) = (0,1,0)$
$f[s_1,s_2]$	3	4	8	$\mu(f[\langle s_1 \rangle, \langle s_2 \rangle]) = (-\frac{1}{2},1,\frac{1}{2})$
$f[s_1,s_3]$	4	5	9	$\mu(f[\langle s_1 \rangle, \langle s_3 \rangle]) = (-1,1,1)$
$f[s_2,s_1]$	3	4	8	$\mu(f[\langle s_2 \rangle, \langle s_1 \rangle]) = (-\frac{1}{2},1,\frac{1}{2})$
$f[s_2,s_2]$	5	6	10	$\mu(f[\langle s_2 \rangle, \langle s_2 \rangle]) = (-\frac{3}{2},1,\frac{3}{2})$
$f[s_2,s_3]$	5	6	10	$\mu(f[\langle s_2 \rangle, \langle s_3 \rangle]) = (-\frac{3}{2},1,\frac{3}{2})$
$f[s_3,s_1]$	4	5	9	$\mu(f[\langle s_3 \rangle, \langle s_1 \rangle]) = (-1,1,1)$
$f[s_3,s_2]$	5	6	10	$\mu(f[\langle s_3 \rangle, \langle s_2 \rangle]) = (-\frac{3}{2},1,\frac{3}{2})$
$f[s_3,s_3]$	6	7	11	$\mu(f[\langle s_3 \rangle, \langle s_3 \rangle]) = (-2,1,2)$

Figure 6: Applying L_*^{wta} to the tree series ψ in Example 3.3

dead. It seems to be an interesting question whether this approach carries over the weighted setting in a reasonable way.

In the case of weighted tree automata, the learners mentioned in Section 3 seem to be the only ones known so far. In contrast, a variety of learning approaches for stochastic string languages and recognizable string series have been proposed in the

literature (see, e.g., [31, 9, 12, 10, 11, 14, 15, 25]). It could be interesting to see whether these approaches extend to stochastic tree languages or tree series as well.

Another question that may be worth studying is whether grammatical inference of tree series is easier if some aspects of the target series are already known, such as the support or the yield of the support.

Acknowledgment

I thank the anonymous referees for their critical comments on the overall contents and focus of this paper as well as for their help regarding details. Hopefully, I have been able to use these comments to the reader's advantage. Furthermore, I want to thank Brink van der Merwe for pointing out errors in the computations in Example 3.4.

References

[1] Adriaans, P. and van Zaanen, M. Computational grammar induction for linguists. *Formal Grammars*, 7:57–68, 2004. Electronic open access journal, see `http://grammars.grlmc.com/special.asp`.

[2] Angluin, D. A note on the number of queries needed to identify regular languages. *Information and Computation*, 51:76–87, 1981.

[3] Angluin, D. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[4] Angluin, D. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[5] Angluin, D. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[6] Becerra-Bonache, L., Dediu, A.H., and Tîrnăucă, C. Learning DFA from correction and equivalence queries. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., and Tomita, E., editors, *8th Intl. Coll. Grammatical Inference: Algorithms and Applications (ICGI'06)*, volume 4201 of Lecture Notes in Computer Science, pages 281–292. Springer, 2006.

[7] Besombes, J. and Marion, J.-Y. Learning tree languages from positive examples and membership queries. *Theoretical Computer Science*, 382:183–197, 2007.

[8] Borchardt, B. The Myhill-Nerode theorem for recognizable tree series. In Ésik, Z. and Fülöp, Z., editors, *Proceedings of the 7th International Conference on Developments in Language Theory (DLT'03)*, volume 2710 of Lecture Notes in Computer Science, pages 146–158. Springer, 2003.

[9] Carrasco, R.C., Forcada, M.L., and Santamaría, L. Inferring stochastic regular grammars with recurrent neural networks. In Higuera, C. de la and Laurent, M., editors, *Proc. 3rd International Colloquium on Grammatical Inference (ICGI'96)*, volume 1147 of Lecture Notes in Artificial Intelligence, pages 274–281. Springer, 1996.

[10] Carrasco, R.C. and Oncina, J. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO Theoretical Informatics and Applications*, 33:1–20, 1999.

[11] Casacuberta, F. and de la Higuera, C. Computational complexity of problems on probabilistic grammars and transducers. In Oliviera, A.L., editor, *Proc. 5th International Colloquium on Grammatical Inference (ICGI'00)*, volume 1891 of Lecture Notes in Artificial Intelligence, pages 15–24. Springer, 2000.

[12] de la Higuera, C. Learning stochastic finite automata from experts. In Honavar, V. and Slutzki, G., editors, *Proc. 4th International Colloquium on Grammatical Inference (ICGI'98)*, volume 1433 of Lecture Notes in Artificial Intelligence, pages 79–89. Springer, 1998.

[13] de la Higuera, C. A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348, 2005.

[14] Denis, F. and Esposito, Y. Learning classes of probabilistic automata. In Shawe-Taylor, J. and Singer, Y., editors, *Proc. 17th Annual Conference on Learning Theory (COLT'04)*, volume 3120 of Lecture Notes in Artificial Intelligence, pages 124–139. Springer, 2004.

[15] Denis, F., Esposito, Y., and Habrard, H. Learning rational stochastic languages. In Lugosi, G. and Simon, H.U., editors, *Proc. 19th Annual Conference on Learning Theory (COLT'06)*, volume 4005 of Lecture Notes in Artificial Intelligence, pages 274–288. Springer, 2006.

[16] Denis, F. and Habrard, H. Learning rational stochastic tree languages. In Hutter, M., Servedio, R.A., and Takimoto, E., editors, *Proc. 18th International Conference on Algorithmic Learning Theory (ALT'07)*, volume 4754 of Lecture Notes in Computer Science, pages 242–256. Springer, 2007.

[17] Drewes, F. and Högberg, J. Extensions of a MAT learner for regular tree languages. In Minock, M.J., Eklund, P., and Lindgren, H., editors, *Proc. 23rd Annual Workshop of the Swedish Artificial Intelligence Society (SAIS'06)*, pages 35–44, 2006.

[18] Drewes, F. and Högberg, J. Query learning of regular tree languages: How to avoid dead states. *Theory of Computing Systems*, 40:163–185, 2007.

[19] Drewes, F. and Vogler, H. Learning deterministically recognizable tree series. *Journal of Automata, Languages and Combinatorics*, 12:333–354, 2007.

[20] Fernau, H. Learning tree languages from text. In Kivinen, J. and Sloan, R. H., editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT'02)*, volume 2375 of Lecture Notes in Artificial Intelligence, pages 153–168. Springer, 2002.

[21] Fernau, H. and de la Higuera, C. Grammar induction: An invitation for formal language theorists. *Formal Grammars*, 7:45–55, 2004. Electronic open access journal, see `http://grammars.grlmc.com/special.asp`.

[22] Fülöp, Z. and Vogler, H. Weighted tree automata and tree transducers. In Kuich, Werner, Droste, Manfred, and Vogler, H., editors, *Handbook of Weighted Automata*. Springer, 2009.

[23] Gold, E.M. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[24] Gold, E.M. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[25] Habrard, H., Denis, F., and Esposito, Y. Using pseudo-stochastic rational languages in probabilistic grammatical inference. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., and Tomita, E., editors, *Proc. 8th International Colloquium on Grammatical Inference (ICGI'06)*, volume 4201 of Lecture Notes in Artificial Intelligence, pages 112–124. Springer, 2006.

[26] Habrard, H. and Oncina, J. Learning multiplicity tree automata. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., and Tomita, E., editors, *Proc. 8th International Colloquium on Grammatical Inference (ICGI'06)*, volume 4201 of Lecture Notes in Artificial Intelligence, pages 268–280. Springer, 2006.

[27] Knuutila, T. and Steinby, M. The inference of tree languages from finite samples: an algebraic approach. *Theoretical Computer Science*, 129:337–367, 1994.

[28] Lee, L. Learning of context-free languages: A survey of the literature. Report TR-12-96, Harvard Univ., Center for Research in Computing Technology, 1996.

[29] López, D., Sempere, J.M., and Garcìa, P. Inference of reversible tree languages. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34:1658–1665, 2004.

[30] Maletti, A. Learning deterministically recognizable tree series – revisited. In Bozapalidis, S. and Rahonis, G., editors, *Proc. 2nd International Conference on Algebraic Informatics (CAI'07)*, volume 4728 of Lecture Notes in Computer Science, pages 218–235. Springer, 2007.

[31] Ron, D., Singer, Y., and Tishby, N. On the learnability and usage of acyclic probabilistic finite automata. In *Proc. 8th Annual Conference on Learning Theory (COLT'95)*, pages 31–40. ACM Press, 1995.

[32] Sakakibara, Y. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76:223–242, 1990.

[33] Sakakibara, Y. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992.

[34] Sakakibara, Y. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1):15–45, 1997.

[35] Shapiro, E.Y. *Algorithmic Program Debugging.* MIT Press, Cambridge, MA, USA, 1983.

[36] Tîrnăucă, C. A note on the relationship between different types of correction queries. In Clark, A., Coste, F., and Miclet, L., editors, *Proc. 9th International Colloquium on Grammatical Inference (ICGI'08)*, volume 5278 of Lecture Notes in Artificial Intelligence, pages 213–223. Springer, 2008.

[37] Tîrnăucă, C. and Kobayashi, S. A characterization of the language classes learnable with correction queries. In Cai, J.-Y., Cooper, S.B., and Zhu, H., editors, *Proc. 4th Annual Conference on Theory and Applications of Models of Computation (TAMC'07)*, volume 4484 of Lecture Notes in Computer Science, pages 398–407. Springer, 2007.

[38] Tîrnăucă, C.I. and Tîrnăucă, C. Learning regular tree languages from correction and equivalence queries. *Journal of Automata, Languages and Combinatorics*, 12:501–524, 2007.

[39] Valiant, L.G. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

Received 11th July 2008

Acta Cybernetica 19 (2009) 275–293.

Weighted Tree-Walking Automata*

Zoltán Fülöp† and Loránd Muzamel†

Abstract

We define weighted tree-walking automata. We show that the class of tree series recognizable by weighted tree-walking automata over a commutative semiring K is a subclass of the class of regular tree series over K: If K is not a ring, then the inclusion is strict.

Keywords: semirings, regular tree series, weighted tree-walking automata

1 Introduction

The concept of a *tree-walking automaton* (for short: twa) was introduced in [1] for modelling syntax-directed translations from strings to strings. Recently its importance grew in XML theory, see, e.g. [23, 25, 26]. A twa A is a sequential finite-state tree acceptor with finitely many transition rules. Obeying its state-behaviour, A walks along the edges of an input tree $s \in T_\Sigma$, where Σ is the input ranked alphabet of A. Then A accepts s if there is an *accepting run on* s, i.e., a finite walk on s from the initial state to the accepting state. Tree languages recognized by twa are effectively regular. Unfortunately there is no straight proof of this fact in the literature, but it can be obtained, e.g., as the special case of the main result of [14]. However, there exists a regular tree language that cannot be recognized by any twa [4]. There are several extensions of twa which still recognize regular tree languages, such as twa with weak pebbles [13], strong pebbles [14], invisible pebbles [15], and also the alternating pebble twa of [24].

Another kind of automata in which we are interested is the weighted tree automaton (for short: wta). It is a natural generalization of the classical tree automaton [10, 19, 20]. The generalization lies in that input trees are supplied with weights taken from an underlying semiring K. In fact, each transition rule of the wta has a weight represented by an element of K. The weight of a run over an input tree $s \in T_\Sigma$ is just the (semiring) product of the transitions which take part

*The full version of a submission presented at *Weighted Automata: Theory and Applications* (Dresden University of Technology, Germany, May 13-16, 2008). The research was supported by the Hungarian Scientific Fund under Grant T 46686 and by the Fund for Teaching and Research in Informatics.

†Department of Foundations of Computer Science, University of Szeged, Árpád tér 2., H-6720 Szeged, Hungary,E-mail: {fulop,muzamel}@inf.u-szeged.hu

in that run. Then, the weight of s is the (semiring) sum of all runs over s. In this way a wta recognizes a tree series, i.e., a mapping from T_Σ to K. Tree series recognizable by wta are called *regular*, and the class of regular tree series which are recognizable by weighted tree automata over Σ and K is denoted by $\mathrm{REG}(\Sigma, K)$. Note that $\mathrm{REG}(\Sigma, \mathbb{B})$, where $\mathbb{B}$ is the Boolean semiring, is the class of recognizable tree languages [19, 20]. Wta were defined and considered in several works, see e.g. [3], [2], [8], [5], [16], [11], [22], and the survey paper [18].

In this paper we introduce the weighted version of a twa, following the idea that led from classical tree automata to wta. In a *weighted tree-walking automaton A (over Σ and K)* (for short: wtwa), every transition rule has a weight taken from the semiring K. We assume that A is non-looping, i.e., it cannot enter into an infinite cycle from the initial configuration. The *weight of a run of A on an input tree $s \in T_\Sigma$* is the product of the weights of the applied transition rules, then the *weight of s computed by A* is the sum of the weights of all the accepting runs of A on s. Since A is non-looping, it has only finitely many such accepting runs. The tree series recognized by A is $S_A : T_\Sigma \to K$, where $S_A(s)$ is the weight of s for every input tree $s \in T_\Sigma$. We denote the class of tree series which are recognizable by non-looping wtwa over Σ and K by $\mathrm{TWA}(\Sigma, K)$. Hence, wtwa with their sequential processing are alternative tools besides the classical weighted tree automata, which process trees parallelly. We note that arbitrary (i.e., maybe looping) wtwa over $\mathbb{B}$ are exactly the twa of [1].

As the main result, we show that if K is commutative, then $\mathrm{TWA}(\Sigma, K) \subseteq \mathrm{REG}(\Sigma, K)$ (Theorem 17). The proof of that the tree series recognized by a non-looping wtwa A is regular is performed according to the following steps. We encode an accepting run of A by annotating the nodes of the input tree by the rules applied in that run. Thus we obtain the concept of a *run tree* of A. Then we construct two twa such that a Boolean combination of the tree languages recognized by them turns out to be the set of run trees of A. Hence, the run trees of A form a regular tree language (Corollary 15). This implies that the tree series S that associates a run tree with the weight of the run it encodes is regular. Finally, we define an appropriate relabeling τ from the set of run trees of A to the set of input trees such that the extension of τ to tree series takes S to S_A. Since such an extension preserves regularity of tree series, we obtain that S_A is regular (Theorem 16).

Then we show that if, in addition, the semiring K is proper, i.e., is not a ring, then the inclusion is strict, i.e., $\mathrm{REG}(\Sigma, K) - \mathrm{TWA}(\Sigma, K) \neq \emptyset$. Hereby we generalize the main result of [4]. We prove this by taking a surjective homomorphism $h : K \to \mathbb{B}$. Now, if $\mathrm{TWA}(\Sigma, K) = \mathrm{REG}(\Sigma, K)$, then also $h(\mathrm{TWA}(\Sigma, K)) = \mathrm{TWA}(\Sigma, \mathbb{B}) = h(\mathrm{REG}(\Sigma, K)) = \mathrm{REG}(\Sigma, \mathbb{B})$ which contradicts the celebrated result $\mathrm{TWA}(\Sigma, \mathbb{B}) \subset \mathrm{REG}(\Sigma, \mathbb{B})$ of [4].

The paper is organized as follows. In Section 2 we introduce the necessary notions and notation. In Section 3 we define weighted tree-walking automata, then we prove our main results in Section 4. In Section 5 we summarize our results and show an open problem.

2 Definitions and notation

2.1 Sets, relations, and strings

We denote the set of nonnegative integers by $\mathbb{N}$. For every $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. The empty set is denoted by $\emptyset$.

For a set A, we denote by 2^A the power set of A and by A^* the set of *strings* over A. We denote *empty string* by ε. Sometimes we write a for a singleton $\{a\}$.

Let $\rho \subseteq A \times A$ be a binary relation. The fact that $(a, b) \in \rho$ for some $a, b \in A$ is also denoted by $a \rho b$. Moreover, the transitive closure and the reflexive, transitive closure are denoted by ρ^+ and ρ^*, respectively.

2.2 Trees and tree languages

A *ranked alphabet* is an ordered pair $(\Sigma, rank)$, where Σ is a finite, nonempty set and $rank$ is a mapping of type $\Sigma \to \mathbb{N}$. For every $k \geq 0$, we define $\Sigma^{(k)} = \{\sigma \in \Sigma \mid rank(\sigma) = k\}$. We define $maxrank(\Sigma) = \max\{rank(\sigma) \mid \sigma \in \Sigma\}$. In the sequel we drop $rank$ and write a ranked alphabet as Σ. Moreover, in the rest of the paper Σ and Δ will denote arbitrary ranked alphabets.

The set of *trees over* Σ *indexed by* A, denoted by $T_\Sigma(A)$, is the smallest set $T \subseteq (\Sigma \cup \{(,)\} \cup \{,\})^*$ such that $\Sigma^{(0)} \cup A \subseteq T$ and whenever $k \geq 1$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T$, then $\sigma(t_1, \ldots, t_k) \in T$. In case $A = \emptyset$, we write T_Σ for $T_\Sigma(A)$. Certainly, $T_\Sigma \neq \emptyset$ if and only if $\Sigma^{(0)} \neq \emptyset$. Every subset $L \subseteq T_\Sigma$ is called a *tree language*.

For every tree $s \in T_\Sigma$, we define the set $pos(s) \subseteq \mathbb{N}^*$ of *the nodes of* s as follows. We let $pos(s) = \{\varepsilon\}$ if $s \in \Sigma^{(0)}$, and $pos(s) = \{\varepsilon\} \cup \{iu \mid 1 \leq i \leq k, u \in pos(s_i)\}$ if $s = \sigma(s_1, \ldots, s_k)$ for some $k \geq 1$, $\sigma \in \Sigma^{(k)}$ and $s_1, \ldots, s_k \in T_\Sigma$.

Now, for a tree $s \in T_\Sigma$ and a node $u \in pos(s)$, *the label of* s *at node* u, denoted by $s(u)$, is defined in a standard way. By the *root of* s we mean the node ε. A node u of s is a *leaf* if $u1 \notin pos(s)$. Moreover, we define the parent of u, denoted by $parent(u)$, and the child number of u, denoted by $childno(u)$, as follows:
(i) if $u = \varepsilon$, then $childno(u) = 0$ and $parent(u)$ is undefined,
(ii) if $u = u'j$, where $u' \in pos(s)$ and $j \in \mathbb{N}$, then $childno(u) = j$ and $parent(u) = u'$.

We will freely use the concepts of a *regular tree language* and a *(finite) tree automaton*. The unfamiliar reader can consult the works [19, 20], and [9] for these concepts. Moreover, we will need the following known closure properties for regular tree languages, see, e.g., Theorem 4.2 of [19].

Proposition 1. *Regular tree languages are closed under Boolean operations.*

2.3 Semirings and tree series

A *semiring* is an algebraic structure $(K, +, \cdot, 0, 1)$ with binary operations addition $+$, multiplication $\cdot$, and constants 0 and 1 (with $0 \neq 1$) such that $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, multiplication distributes over addition (both from left and right), and $a \cdot 0 = 0 \cdot a = 0$ for every $a \in K$. Frequently we

will write just K for $(K, +, \cdot, 0, 1)$. We say that K is *commutative* if $a \cdot b = b \cdot a$ for each $a, b \in K$. The semiring K is *proper* if it is not a ring, i.e., there is no additive inverse of 1.

Examples of commutative semirings are the *Boolean semiring* $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$ and the *arctic semiring* $\mathsf{Arct} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$, where the operations max and $+$ are extended to $\mathbb{N} \cup \{-\infty\}$ in the obvious way. Note that both $\mathbb{B}$ and Arct are proper.

A *tree series (over Σ and K)* is a mapping $S : T_\Sigma \to K$ where (S, s) is usually written rather than $S(s)$ for $s \in T_\Sigma$. We denote by $K\langle\langle T_\Sigma \rangle\rangle$ the set of tree series over Σ and K. Now we give an example of a tree series over Arct.

Example 2. Let Σ be such that $\Sigma^{(0)} = \{\alpha, \beta\}$. The tree series $height_\alpha \in \mathsf{Arct}\langle\langle T_\Sigma \rangle\rangle$ delivers, for $s \in T_\Sigma$, the length of the longest path from the root of s to an α-node. More exactly,

(i) $(height_\alpha, s) = \begin{cases} 0 & \text{if } s = \alpha \\ -\infty & \text{if } s = \beta, \end{cases}$

(ii) $(height_\alpha, s) = 1 + \max\{(height_\alpha, s_i) \mid 1 \le i \le k\}$ if $s = \sigma(s_1, \ldots, s_k)$ for some $k \ge 1$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T_\Sigma$.

In an analogous way, we can define $height_\beta \in \mathsf{Arct}\langle\langle T_\Sigma \rangle\rangle$. Now by an α-β-path in s we mean a path from an α-node to a β-node of s along the edges of s such that the α-node precedes the β-node in the usual lexicographical order of the nodes and that every edge is taken at most once in the path. Finally, we define the tree series $width_{\alpha\beta} \in \mathsf{Arct}\langle\langle T_\Sigma \rangle\rangle$ which delivers, for $s \in T_\Sigma$, the length of the longest α-β-path in s. More exactly,

(i) $(width_{\alpha\beta}, s) = -\infty$ if $s = \alpha$ or $s = \beta$,

(ii) $(width_{\alpha\beta}, s) = \max\{ \max\{(height_\alpha, s_i) + (height_\beta, s_j) + 2 \mid 1 \le i < j \le k\}, \max\{(width_{\alpha\beta}, s_i) \mid 1 \le i \le k\}\}$

if $s = \sigma(s_1, \ldots, s_k)$ for some $k \ge 1$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T_\Sigma$.

In particular, let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}, \beta^{(0)}\}$. Then, for the tree $s = \sigma(\sigma(\sigma(\sigma(\beta, \alpha), \beta), \gamma(\sigma(\beta, \alpha))), \beta)$, we have $height_\alpha(s) = 4$, $height_\beta(s) = 4$, and $width_{\alpha\beta}(s) = 6$. The longest α-β-path in s is visualized in Fig. 1.

Weighted tree automata were defined in several works in several ways, see e.g. [3], [2], [8], [5], [16], [11], [22], and the survey paper [18]. Here we give a definition which is equivalent with the standard one and, at the same time, is easy to handle.

A *weighted tree automaton (wta) over K* is a system $M = (Q, \Sigma, R, \nu)$, where Q is a finite set of *states*, Σ is the *input ranked alphabet*, ν is a mapping of type $Q \to K$, and R is a finite set of *(weighted) rules* of the form $\sigma(q_1, \ldots, q_k) \xrightarrow{a} q$, where $k \ge 0$, $\sigma \in \Sigma^{(k)}$, $q_1, \ldots, q_k, q \in Q$, and $a \in K$. In case $k = 0$ we write $\sigma \xrightarrow{a} q$ rather than $\sigma() \xrightarrow{a} q$.

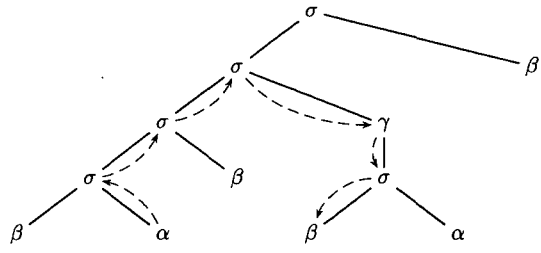


Figure 1: The longest α-β-path in the tree $s = \sigma(\sigma(\sigma(\sigma(\beta,\alpha),\beta),\gamma(\sigma(\beta,\alpha))),\beta)$.

The tree series $S_{M,q} \in K\langle\langle T_\Sigma \rangle\rangle$ recognized by M in a state $q \in Q$ is defined as follows. For every input tree $s = \sigma(s_1,\ldots,s_k) \in T_\Sigma$ with $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $s_1,\ldots,s_k \in T_\Sigma$, we have

$$(S_{M,q}, s) = \sum_{\substack{q_1,\ldots,q_k \in Q,\ a \in K \\ \sigma(q_1,\ldots,q_k) \xrightarrow{a} q \in R}} (S_{M,q_1}, s_1) \cdot \ldots \cdot (S_{M,q_k}, s_k) \cdot a.$$

Moreover, the tree series *recognized by* M is defined by

$$(S_M, s) = \sum_{q \in Q} (S_{M,q}, s) \cdot \nu(q)$$

for every input tree $s \in T_\Sigma$.

A tree series is *regular* if it can be recognized by a weighted tree automaton. We denote the class of regular tree series over Σ and K by $\mathrm{REG}(\Sigma, K)$.

Next we define the concept of a characteristic tree series and some operations on tree series which we need in the sequel. For a tree language $L \subseteq T_\Sigma$ the *characteristic tree series of* L is $\mathcal{X}_L \in K\langle\langle T_\Sigma \rangle\rangle$, defined by $(\mathcal{X}_L, s) = 1$ if $s \in L$ and $(\mathcal{X}_L, s) = 0$ otherwise for every $s \in T_\Sigma$.

Let $S_1, S_2 \in K\langle\langle T_\Sigma \rangle\rangle$ be tree series and $a \in K$. We define the tree series $aS_1 \in K\langle\langle T_\Sigma \rangle\rangle$ by $(aS_1, s) = a \cdot (S_1, s)$ for every $s \in T_\Sigma$. The *sum* and the *Hadamard product of* S_1 *and* S_2 are denoted by $S_1 + S_2$ and $S_1 \odot S_2$ in $K\langle\langle T_\Sigma \rangle\rangle$, respectively, and are defined as $(S_1 + S_2, s) = (S_1, s) + (S_2, s)$ and $(S_1 \odot S_2, s) = (S_1, s) \cdot (S_2, s)$ for each $s \in T_\Sigma$.

Regular tree series have the following closure properties.

Proposition 3. *Let K be commutative.*

(a) If $L \subseteq T_\Sigma$ is a regular tree language, then $\mathcal{X}_L \in \mathrm{REG}(\Sigma, K)$.

(b) If $S_1, S_2 \in \mathrm{REG}(\Sigma, K)$ and $a \in K$, then aS_1, $S_1 + S_2$, and $S_1 \odot S_2$ are also in $\mathrm{REG}(\Sigma, K)$.

For the proof of (a) we refer the reader to Lemma 3.3 of [12]. The closure under the multiplication with a, the sum, and the Hadamard product were proved in Lemmata 6.3 and 6.4 of [11] (cf. also Lemma 3.3 of [12]), and in Corollary 3.9

of [6]. Let us note that the proof of (a) and of the closure under sum do not need commutativity of K.

Now we define the concept of relabeling. A (deterministic) *relabeling* is a mapping $\tau : \Sigma \to \Delta$ such that for each $k \geq 0$ and $\sigma \in \Sigma^{(k)}$ we have $\tau(\sigma) \in \Delta^{(k)}$. Then τ extends to the mapping $\tau' : T_\Sigma \to T_\Delta$, where $\tau'(s) = \tau(\sigma)(\tau'(s_1), \ldots, \tau'(s_k))$ for every $s = \sigma(s_1, \ldots, s_k)$ with $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T_\Sigma$. For each tree $t \in T_\Delta$, the set $\{s \in T_\Sigma \mid t = \tau'(s)\}$ is finite. Finally, τ' (and hence τ) extends to the mapping $\hat{\tau} : K\langle\langle T_\Sigma \rangle\rangle \to K\langle\langle T_\Delta \rangle\rangle$, where

$$(\hat{\tau}(S), t) = \sum_{s \in T_\Sigma,\, \tau'(s) = t} (S, s).$$

for each $S \in K\langle\langle T_\Sigma \rangle\rangle$ and $t \in T_\Delta$.

We will need the following result which was proved in Lemma 3.4 of [12].

Proposition 4. *Let K be commutative. If $S \in \mathrm{REC}(\Sigma, K)$ and $\tau : \Sigma \to \Delta$ is a relabeling, then $\hat{\tau}(S) \in \mathrm{REC}(\Delta, K)$.*

Now let K' be another semiring and $h : K \to K'$ a semiring homomorphism. Then h extends to the mapping $h : K\langle\langle T_\Sigma \rangle\rangle \to K'\langle\langle T_\Sigma \rangle\rangle$ by defining $h(S) = h \circ S$ for every $S \in K\langle\langle T_\Sigma \rangle\rangle$. We will need the following result, cf. Lemma 3 of [7] and Theorem 3.9 of [18].

Proposition 5. *(a) For every $S \in \mathrm{REC}(\Sigma, K)$ and semiring homomorphism $h : K \to K'$, we have $h(S) \in \mathrm{REC}(\Sigma, K')$, i.e., $h(\mathrm{REC}(\Sigma, K)) \subseteq \mathrm{REC}(\Sigma, K')$. (b) If, in addition, h is surjective, then $h(\mathrm{REC}(\Sigma, K)) = \mathrm{REC}(\Sigma, K')$.*

3 Weighted tree-walking automata

In this section we define weighted tree-walking automata. For the definition of the classical (unweighted) case, see [1] and [13].

Informally, a weighted tree-walking automaton A over a semiring K works as follows on input tree s. It is equipped with a pointer that walks on the edges of s, obeying its state behaviour. In a given moment of the computation, the current node is the node of s pointed by the pointer. Each computation step is determined by the state of the given moment, the label, and the child number of the current node. Besides, each computation step has a weight over K. An accepting run of A on s is a walk starting at the root of s in the initial state and finishing at an arbitrary node of s in the (only) accepting state. The weight of an accepting run is the product of the weights of the corresponding computation steps. Finally, the weight of s, computed by A, is the sum of the weights of the accepting runs of A on s. Now we give the exact definition.

Syntax

For each $\sigma \in \Sigma$, the set of *instructions determined by* σ is the set

$$I_{\sigma,j} = \begin{cases} \{stay, up, down_i \mid 1 \leq i \leq rank(\sigma)\} & \text{if } j > 0, \\ \{stay, down_i \mid 1 \leq i \leq rank(\sigma)\} & \text{if } j = 0. \end{cases}$$

Let K be a semiring. A *weighted tree-walking automaton* (shortly: *wtwa*) *over* K is a system $A = (Q, \Sigma, q_0, q_a, P)$, where

- Q is a finite set, the *set of states*,
- Σ is the *input ranked alphabet*,
- $q_0, q_a \in Q$ are the *initial* and *accepting state*, respectively, such that $q_0 \neq q_a$, and
- P is a finite *set of rules* of the form $\langle q, \sigma, j\rangle \xrightarrow{a} \langle q', \varphi\rangle$, where $q \in Q - \{q_a\}$, $\sigma \in \Sigma$, $j \in \{0, \ldots, maxrank(\Sigma)\}$, $a \in K$ such that $a \neq 0$, $q' \in Q - \{q_0\}$ and $\varphi \in I_{\sigma,j}$.

A rule $\pi = \langle q, \sigma, j\rangle \xrightarrow{a} \langle q', \varphi\rangle$ is called both a q-rule and a σ-rule. The left-state, right-state, and the weight of π are defined by $lstate(\pi) = q$, $rstate(\pi) = q'$, and $wt(\pi) = a$, respectively. Moreover, we let $test(\pi) = (\sigma, j)$. Rules with left-state q_0 (right-state q_a) are called *initial rules* (*accepting rules*).

Computation relation

Let $s \in T_\Sigma$ be an input tree. For a node $u \in pos(s)$, we define $test_s(u) = (\sigma, j)$, where $\sigma = s(u)$ and $j = childno(u)$. If s is clear from the context, then we will write $test(u)$ for $test_s(u)$. Now assume that $test(u) = (\sigma, j)$ and let $\varphi \in I_{\sigma,j}$ be an instruction. Then we define the effect of φ on u by

$$\varphi(u) = \begin{cases} u & \text{if } \varphi = stay, \\ parent(u) & \text{if } \varphi = up, \\ ui & \text{if } \varphi = down_i. \end{cases}$$

A *configuration* (of A over s) is a tuple $\langle q, u\rangle$, where $q \in Q$ and $u \in pos(s)$. We denote the set of configurations of A over s by $C_{A,s}$ and write just C_s if A is clear from the context. In particular, $\langle q_0, \varepsilon\rangle$ is the *initial configuration* and $\langle q_a, u\rangle$ is an *accepting configuration* for every $u \in pos(s)$.

Let $\pi \in P$ be a rule. The *π-transition relation (of A with respect to s)* is the binary relation $\vdash_{A,s,\pi}$ over C_s defined as follows. For arbitrary configurations $\langle q, u\rangle$, $\langle q', u'\rangle \in C_s$ we have $\langle q, u\rangle \vdash_{A,s,\pi} \langle q', u'\rangle$ if and only if

- π has the form $\langle q, \sigma, j\rangle \xrightarrow{a} \langle q', \varphi\rangle$,
- $test(u) = (\sigma, j)$, and $u' = \varphi(u)$.

If A is clear from the context, then we write $\vdash_{s,\pi}$ for $\vdash_{A,s,\pi}$. Finally, we define the *transition relation* $\vdash_s$ *(of A with respect to s)* as $\vdash_s = \bigcup_{\pi \in P} \vdash_{s,\pi}$.

Looping property

We define A to be *looping* if there is an input tree $s \in T_\Sigma$ and a configuration $\langle q, u\rangle \in C_s$ such that $\langle q_0, \varepsilon\rangle \vdash_s^* \langle q, u\rangle \vdash_s^+ \langle q, u\rangle$. Otherwise, A is *non-looping*. We call $\langle q, u\rangle$ a *looping configuration*. The looping problem of wtwa is decidable. In fact, the decidability of the more general *circularity problem* is proved in a more general setting for pebble macro tree transducers, cf. Section 4. of [17].

Accepting runs

Let $s \in T_\Sigma$ be an input tree. An *accepting run (of A on s)* is a string

$$r = \langle q_0, u_0\rangle \pi_0 \langle q_1, u_1\rangle \pi_1 \dots \pi_k \langle q_{k+1}, u_{k+1}\rangle$$

over $C_s \cup P$, where $k \geq 0$, $\langle q_0, u_0\rangle, \dots, \langle q_{k+1}, u_{k+1}\rangle \in C_s$ with $u_0 = \varepsilon$ and $q_{k+1} = q_a$, $\pi_0, \dots, \pi_k \in P$, and $\langle q_i, u_i\rangle \vdash_{s,\pi_i} \langle q_{i+1}, u_{i+1}\rangle$ for each $0 \leq i \leq k$. The *weight of r* is $wt(r) = wt(\pi_0) \cdot \dots \cdot wt(\pi_k)$. We denote the set of accepting runs of A on s by $Accrun_s$.

Tree series recognized by non-looping wtwa

Let A be a non-looping wtwa. The tree series $S_A \in K\langle\langle T_\Sigma\rangle\rangle$ *recognized by* A is defined as

$$(S_A, s) = \sum_{r \in Accrun_s} wt(r).$$

for each $s \in T_\Sigma$. Note that we need non-looping property to guarantee that the set $Accrun_s$ is finite and hereby the above sum has finitely many members. We denote by $\mathrm{TWA}(\Sigma, K)$ the class of tree series that are recognizable by non-looping wtwa over Σ and K.

Example

Next we give an example of wtwa which recognizes the tree series $width_{\alpha\beta}$ defined in Example 2.

Example 6. Let $A = (\{q_0, q_a, q_1, q_2, q_{up}^{(l)}, q_{up}^{(r)}\}, \Sigma, q_0, q_a, P)$ be a wtwa over Arct, where Σ is the particular alphabet in Example 2 and P is the set of the following rules.

Initial rules:

$$\pi_1 : \langle q_0, \sigma, 0\rangle \xrightarrow{0} \langle q_1, down_1\rangle \quad \pi_2 : \langle q_0, \sigma, 0\rangle \xrightarrow{0} \langle q_1, down_2\rangle$$
$$\pi_3 : \langle q_0, \gamma, 0\rangle \xrightarrow{0} \langle q_1, down_1\rangle$$

Intermediate rules:

$$\pi_4 : \langle q_1, \sigma, 1\rangle \xrightarrow{0} \langle q_1, down_1\rangle \qquad \pi_{15} : \langle q_{up}^{(l)}, \sigma, 1\rangle \xrightarrow{1} \langle q_{up}^{(l)}, up\rangle$$
$$\pi_5 : \langle q_1, \sigma, 1\rangle \xrightarrow{0} \langle q_1, down_2\rangle \qquad \pi_{16} : \langle q_{up}^{(l)}, \sigma, 2\rangle \xrightarrow{1} \langle q_{up}^{(r)}, up\rangle$$
$$\pi_6 : \langle q_1, \sigma, 2\rangle \xrightarrow{0} \langle q_1, down_1\rangle \qquad \pi_{17} : \langle q_{up}^{(r)}, \sigma, 1\rangle \xrightarrow{1} \langle q_{up}^{(l)}, up\rangle$$
$$\pi_7 : \langle q_1, \sigma, 2\rangle \xrightarrow{0} \langle q_1, down_2\rangle \qquad \pi_{18} : \langle q_{up}^{(r)}, \sigma, 2\rangle \xrightarrow{1} \langle q_{up}^{(r)}, up\rangle$$
$$\pi_8 : \langle q_1, \gamma, 1\rangle \xrightarrow{0} \langle q_1, down_1\rangle \qquad \pi_{19} : \langle q_{up}^{(l)}, \gamma, 1\rangle \xrightarrow{1} \langle q_{up}^{(l)}, up\rangle$$
$$\pi_9 : \langle q_1, \gamma, 2\rangle \xrightarrow{0} \langle q_1, down_1\rangle \qquad \pi_{20} : \langle q_{up}^{(l)}, \gamma, 2\rangle \xrightarrow{1} \langle q_{up}^{(r)}, up\rangle$$
$$\pi_{10} : \langle q_1, \alpha, 1\rangle \xrightarrow{1} \langle q_{up}^{(l)}, up\rangle \qquad \pi_{21} : \langle q_2, \sigma, 1\rangle \xrightarrow{1} \langle q_2, down_1\rangle$$
$$\pi_{11} : \langle q_1, \alpha, 2\rangle \xrightarrow{1} \langle q_{up}^{(r)}, up\rangle \qquad \pi_{22} : \langle q_2, \sigma, 1\rangle \xrightarrow{1} \langle q_2, down_2\rangle$$
$$\pi_{12} : \langle q_{up}^{(l)}, \sigma, 0\rangle \xrightarrow{1} \langle q_2, down_2\rangle \qquad \pi_{23} : \langle q_2, \sigma, 2\rangle \xrightarrow{1} \langle q_2, down_1\rangle$$
$$\pi_{13} : \langle q_{up}^{(l)}, \sigma, 1\rangle \xrightarrow{1} \langle q_2, down_2\rangle \qquad \pi_{24} : \langle q_2, \sigma, 2\rangle \xrightarrow{1} \langle q_2, down_2\rangle$$
$$\pi_{14} : \langle q_{up}^{(l)}, \sigma, 2\rangle \xrightarrow{1} \langle q_2, down_2\rangle \qquad \pi_{25} : \langle q_2, \gamma, 1\rangle \xrightarrow{1} \langle q_2, down_1\rangle$$
$$\pi_{26} : \langle q_2, \gamma, 2\rangle \xrightarrow{1} \langle q_2, down_1\rangle$$

Accepting rules:

$$\pi_{27} : \langle q_2, \beta, 1\rangle \xrightarrow{0} \langle q_a, stay\rangle \qquad \pi_{28} : \langle q_2, \beta, 2\rangle \xrightarrow{0} \langle q_a, stay\rangle$$

Note that A is non-looping. Moreover, A works on an input tree s as follows.

1) In the first phase, A moves the pointer nondeterministically to an α-node of s with its rules π_1-π_9. The weights of these steps are 0.

2) Then A moves its pointer upwards (using π_{10} and π_{11}), such that states $q_{up}^{(l)}$ and $q_{up}^{(r)}$ store whether the previous step was made from the left child or the right child, respectively. If A is in state $q_{up}^{(l)}$ and the pointed node is labeled by σ, then A nondeterministically decides whether to continue moving up (using $\pi_{15} - \pi_{20}$) or to move down to the second child in state q_2 (using $\pi_{12} - \pi_{14}$). Each of these steps has weight 1. (We do not need γ-rules with left-state $q_{up}^{(r)}$ because going up to a node labelled by γ leads to state $q_{up}^{(l)}$.)

3) In the third phase, A searches nondeterministically for a β-node of s by descending with its rules π_{21}-π_{26}. Each of these steps has weight 1. If it finds a β-node, then the run terminates in the accepting state q_a with the 0-weighted rules π_{27} or π_{28}.

Now it is easy to see that an accepting run r of A on s contains an α-β path in s and that the weight of r is the length of that α-β path. Hence we conclude that, for each input tree $s \in T_\Sigma$, the wtwa A computes $(width_{\alpha\beta}, s)$ and thus it recognizes the tree series $width_{\alpha\beta}$.

An accepting run of A on the input tree s of Example 2 is

$$r = \langle q_0, \varepsilon\rangle \pi_1 \langle q_1, 1\rangle \pi_4 \langle q_1, 11\rangle \pi_4 \langle q_1, 111\rangle \pi_5 \langle q_1, 1112\rangle \pi_{11} \langle q_{up}^{(r)}, 111\rangle \pi_{17}$$
$$\langle q_{up}^{(l)}, 11\rangle \pi_{15} \langle q_{up}^{(l)}, 1\rangle \pi_{13} \langle q_2, 12\rangle \pi_{26} \langle q_2, 121\rangle \pi_{21} \langle q_2, 1211\rangle \pi_{27} \langle q_a, 1211\rangle. \quad (3.1)$$

In fact, r contains the longest α-β-path in s, which can be seen in Fig. 1.

Tree-walking automata as the Boolean case of wtwa

Later in the paper we will consider wtwa over $\mathbb{B}$. For such a wtwa A, the weight of each rule and hence of each accepting run is 1. Moreover, since 1+1=1 in $\mathbb{B}$, we do not need the restriction that A is non-looping to define the semantics of A. Hence, we define S_A for an (arbitrary) wtwa A over $\mathbb{B}$ by the formula used for a non-loping one. It is easily seen that $(S_A, s) = 1$, i.e., A *accepts* s, if and only if there is an accepting run of A on the input tree s (even in the case that A has infinitely many accepting runs on s, i.e., that $Accrun_s$ is infinite). In this way A can also be considered as a *tree recognizer* and we obtain the *tree-walking automata (twa)* of [1], cf. also [13]. In fact, we call a wtwa A over $\mathbb{B}$ a twa, we drop the weight 1 from the specification of its rules, and we denote by $L(A)$ the set of trees accepted by A and call it the tree language *recognized by* A.

Twa with one pebble

We will also need a more general tree recognizer, the so called *one pebble tree-walking automata (1-ptwa)*, see e.g. [13].

A 1-ptwa A works as follows on input tree s. Similarly to a twa, A is equipped with a pointer that walks on the edges of s. Moreover, A has one pebble that is able to mark a node of s, i.e., that can be dropped at and lifted from the current node. After marking a node by the pebble, A can walk away from the marked node. When accessing a node, A is able to test whether the current node is marked by the pebble or not and the further computation of A may depend on the result of this test. This gives an extra computation power for A. Like a twa, A will accept s if and only if it has at least one accepting run on s, and the set of trees accepted by A is denoted by $L(A)$. Again, we call $L(A)$ the tree series *recognized by* A. For a formal definition of a 1-ptwa, the reader is advised to consult [13] or [14].

The tree languages recognized by 1-ptwa (and hence by twa) are effectively regular, which is proved in [13] for ptwa and also in [24] for the more general *pebble alternating tree-walking automata*. This yields the following proposition.

Proposition 7. *The tree languages recognized by 1-ptwa (and hence by twa) are effectively regular.*

4 The recognizing power of non-looping wtwa

We can prove our main results for non-looping wtwa over commutative semirings.

> *Therefore in the rest of this paper K denotes a commutative semiring and $A = (Q, \Sigma, q_0, q_a, P)$ denotes a non-looping wtwa over K.*

We will prove that the tree series recognized by A is effectively regular. For this, we will consider annotated input trees. These are trees the σ-nodes of which are annotated by sets of σ-rules in P, where $\sigma \in \Sigma$. The annotation of a node by a set of rules intuitively means that those rules may be applied at that node. Since A is

non-looping, it will be sufficient to consider consistent annotations. More exactly, let $P(\sigma)$ be the set of all σ-rules in P and define a pair $\langle \sigma, J \rangle \in \Sigma \times P(\sigma)$ with $J = \{\pi_1, \ldots, \pi_m\}$ to be *consistent* if the following conditions hold:

a) $test(\pi_1) = \ldots = test(\pi_m)$ and

b) the states $lstate(\pi_1), \ldots, lstate(\pi_m)$ are pairwise different.

Then, we introduce the ranked alphabet $\Sigma(P) \subseteq \Sigma \times P(\sigma)$ such that

$$\Sigma(P)^{(k)} = \{\langle \sigma, J \rangle \in \Sigma^{(k)} \times P(\sigma) \mid \langle \sigma, J \rangle \text{ is consistent}\}$$

for every $k \geq 0$. Finally we define $rules(\langle \sigma, J \rangle) = J$.

It is useful to introduce the tree series $weight \in K\langle\langle T_{\Sigma(P)} \rangle\rangle$ which associates an annotated tree $t \in T_{\Sigma(P)}$ with the product of the weights of the rules appearing in t. More exactly, for every $t \in T_{\Sigma(P)}$, we have

$$(weight, t) = \prod_{\substack{u \in pos(t) \\ \pi \in rules(t(u))}} wt(\pi),$$

where the empty product yields weight 1. We will need the following result.

Lemma 8. *The tree series weight is regular.*

Proof. Let $M = (\{q\}, \Sigma(P), R, \nu)$ be a wta such that $\nu(q) = 1$ and R the smallest set containing all rules $\langle \sigma, J \rangle(q, \ldots, q) \xrightarrow{a} q$ with $\langle \sigma, J \rangle \in \Sigma(P)$ such that $a = \prod_{\pi \in J} wt(\pi)$. It is easy to see that $S_M = weight$. □

In particular, we will be interested in those annotated trees which encode an accepting run. We call such trees run trees and define them in the following way.

Let $s \in T_\Sigma$ be an input tree and

$$r = \langle q_0, u_0 \rangle \pi_0 \langle q_1, u_1 \rangle \pi_1 \ldots \langle q_k, u_k \rangle \pi_k \langle q_a, u_{k+1} \rangle$$

an accepting run on s. The *run tree of s and r* is the tree $rtree(s, r) \in T_{\Sigma \times 2^P}$ defined by the following conditions:

- $pos(rtree(s, r)) = pos(s)$ and
- for each $u \in pos(rtree(s, r))$ we have

$$rtree(s, r)(u) = \langle s(u), \{\pi_i \in P \mid 0 \leq i \leq k,\ u_i = u\} \rangle.$$

We say that $rtree(s, r)$ *encodes* r. Moreover, the *set of run trees of s* is $Rtree_s = \{rtree(s, r) \mid r \in Accrun_s\}$. In Fig. 2 we visualize the run tree $rtree(s, r)$, where s is the input tree of Example 2 and r is the run of A on s appearing in Example 6.

We can prove easily that run trees are in fact trees over $\Sigma(P)$.

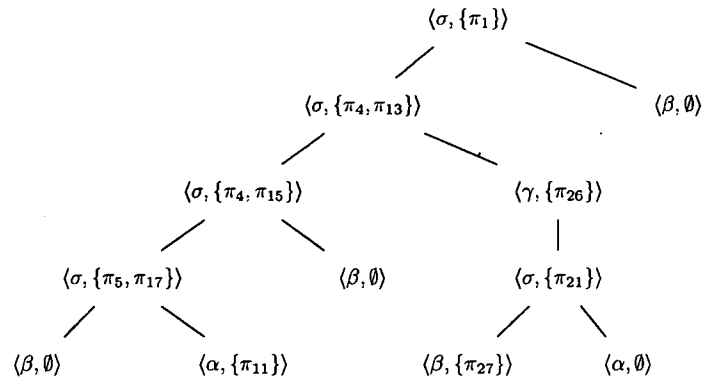


Figure 2: A run tree.

Lemma 9. *For every $s \in T_\Sigma$, we have $Rtree_s \subseteq T_{\Sigma(P)}$.*

Proof. Let $t \in Rtree_s$ be a run tree and $u \in pos(t)$ $(= pos(s))$ a node. Assume that $t(u) = \langle \sigma, \{\pi_1, \ldots, \pi_m\}\rangle$, where $\pi_1, \ldots, \pi_m \in P$. We show that both a) and b) of the definition of consistency hold. Since $t \in Rtree_s$, there is an accepting run $r \in Accrun_s$ such that $t = rtree(s, r)$.

Condition a) obviously holds because a rule π can be applied at a node u of s only if $test(\pi) = test(u)$.

We prove b) by contradiction. Assume that there are $q \in Q$, μ, and ν such that $1 \leq \mu \neq \nu \leq m$ and $q = lstate(\pi_\mu) = lstate(\pi_\nu)$. Then it directly follows that configuration $\langle q, u\rangle$ occurs twice in the accepting run r, i.e., $\langle q_0, \varepsilon\rangle \vdash^*_{A,s} \langle q, u\rangle \vdash^+_{A,s} \langle q, u\rangle$. This contradicts the fact that A is non-looping. □

We will also need the following straightforward result.

Lemma 10. *Let $s \in T_\Sigma$ be an input tree. For each accepting run $r \in Accrun_s$ we have $wt(r) = (weight, rtree(s, r))$.*

Next we show that $Accrun_s$ and $Rtree_s$ have the same number of elements. For this, we define the mapping $\theta_s : Accrun_s \to Rtree_s$ such that $\theta_s(r) = rtree(s, r)$ for each $r \in Accrun_s$ and show that it is a bijection.

Lemma 11. *The mapping θ_s is a bijection.*

Proof. It should be clear that θ_s is surjective. To show that it is injective, we consider $r_1, r_2 \in Accrun_s$ with $rtree(s, r_1) = rtree(s, r_2)$ and show that $r_1 = r_2$. Let

$$r_1 = \langle q_0, u_0\rangle \pi_0 \langle q_1, u_1\rangle \pi_1 \ldots \langle q_k, u_k\rangle \pi_k \langle q_a, u_{k+1}\rangle$$

and

$$r_2 = \langle q_0, u'_0\rangle \pi'_0 \langle q'_1, u'_1\rangle \pi'_1 \ldots \langle q'_l, u'_l\rangle \pi'_l \langle q_a, u'_{l+1}\rangle,$$

where $u_0 = u'_0 = \varepsilon$.

First we prove by contradiction that $\pi_i = \pi'_i$ for every $0 \leq i \leq \min\{k, l\}$. Assume that there is an i such that $\pi_i \neq \pi'_i$ and $\pi_j = \pi'_j$ for every $0 \leq j < i$. The latter implies $q_j = q'_j$ and $u_j = u'_j$ for every $0 \leq j \leq i$. Since $\pi_i \in \mathit{rules}(\mathit{rtree}(s, r_2)(u_i))$ and, in particular, $\mathit{rtree}(s, r_1)(u_i) = \mathit{rtree}(s, r_2)(u_i)$, there is an $i < m \leq l$ such that $u'_m = u_i$ and $\pi'_m = \pi_i$. If $i = 0$, this is a contradiction because the left-state of π'_m cannot be q_0. If $0 < i$, then we get

$$\langle q_i, u_i \rangle = \langle q'_i, u'_i \rangle \vdash_s^+ \langle q'_m, u'_m \rangle = \langle q_i, u_i \rangle,$$

which is a contradiction again because A is non-looping. Hence the statement follows.

This statement and the fact that there are no rules with left-state q_a imply that $k = l$. From the latter $u_{k+1} = u'_{k+1}$ follows, hence $r_1 = r_2$. □

Thus we obtain the following, which we need later.

Corollary 12. *For each input tree $s \in T_\Sigma$ we have* $(S_A, s) = \sum_{t \in \mathit{Rtree}_s} (\mathit{weight}, t)$.

Proof.

$$\begin{aligned}
(S_A, s) &= \sum_{r \in \mathit{Accrun}_s} \mathit{wt}(r) \\
&= \sum_{r \in \mathit{Accrun}_s} (\mathit{weight}, \mathit{rtree}(s, r)) \quad \text{(by Lemma 10)} \\
&= \sum_{t \in \mathit{Rtree}_s} (\mathit{weight}, t) \quad \text{(by Lemma 11).}
\end{aligned}$$

□

The *set of run trees of* A is $\mathit{Rtree}_A = \bigcup_{s \in T_\Sigma} \mathit{Rtree}_s$.

We will show that Rtree_A is a regular tree language. In fact, we will construct a twa A' and a 1-ptwa A'' and then show that Rtree_A is the Boolean combination $L(A') \cap \overline{L(A'')}$ of the tree languages recognized by them.

The twa A' works on trees over $\Sigma(P)$ and accepts all the run trees of A, i.e., $\mathit{Rtree}_A \subseteq L(A')$. Let $A' = (Q, \Sigma(P), q_0, q_a, P')$, where, for every $\langle \sigma, J \rangle \in \Sigma(P)$, the set P' contains the rule

$$\langle q, \langle \sigma, J \rangle, j \rangle \to \langle q', \varphi \rangle$$

if and only if the rule $\langle q, \sigma, j \rangle \to \langle q', \varphi \rangle$ is in J. Note that A' is deterministic in the sense that, due to condition b) of the definition of consistency, it has no different rules with the same left-hand side.

Let us consider an input tree $t \in T_{\Sigma(P)}$ to A' and let $s \in T_\Sigma$ be the tree obtained from t by dropping the rule sets from its labels. It should be clear that A' simulates on t the steps of A on s which apply the rules in the nodes of t in a state-to-state and node-to-node manner. Hence A' accepts t if and only if A accepts s applying the rules in the nodes of t. In particular, for every $s \in T_\Sigma$, the twa A' accepts each $t \in \mathit{Rtree}_s$ because such a t encodes an accepting run of s. (Here we use Lemma 9.)

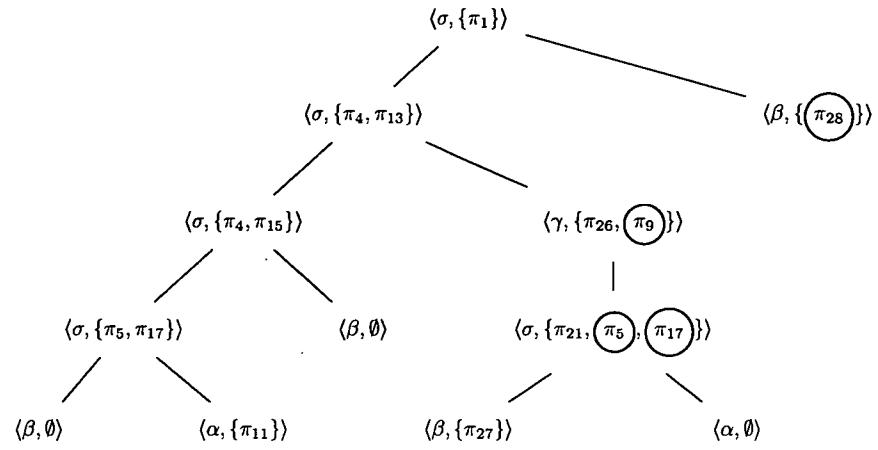


Figure 3: A tree accepted by A'. The circled rules are superfluous.

Hence $Rtree_A \subseteq L(A')$. Note that A' may also accept trees which do not encode accepting runs because they contain some "superfluous rules" in their labels.

In Fig. 3 we show a tree which is accepted by A' but is not a run tree of A, where A is now the wtwa of Example 6. At the same time, the tree contains the accepting run shown in Fig. 2.

In order to clarify the relation between $Rtree_A$ and $L(A')$, we define the concept of the superfluous rule in an exact way.

Let $t \in L(A')$ be a tree and $u \in pos(t)$. A rule $\pi \in rules(t(u))$ *is superfluous at node* u *(in* t*)* if the tree t' is also in $L(A')$, where t' is obtained from t by dropping π from the set $rules(t(u))$. If this is the case, then we say that t contains a superfluous rule.

Lemma 13. $Rtree_A = \{t \in L(A') \mid t$ *contains no superfluous rules*$\}$.

Proof. If $t \in Rtree_A$, then $t \in L(A')$. We show that t contains no superfluous rules by contradiction. For this, let $s \in T_\Sigma$ and $r \in Rtree_s$ be such that $t = rtree(s, r)$. Consider an accepting run

$$r' = \langle q_0, u_0\rangle \Pi_0 \langle q_1, u_1\rangle \Pi_1 \ldots \langle q_k, u_k\rangle \Pi_k \langle q_a, u_{k+1}\rangle$$

of A' on t and let π_i be the rule of A corresponding to Π_i, $1 \leq i \leq n$, see the definition of A' above. It should be clear that the sequence obtained from r' by replacing Π_i with π_i for every $1 \leq i \leq n$ is the accepting run r of A on s. Assume that t contains a superfluous rule. Since r contains all rules in the nodes of t there is an index i such that π_i is superfluous at u_i. Assume that i is minimal. Since π_0 is the only initial rule in r, we have $i > 0$. Let t' be the tree obtained from t by dropping π_i (whose left-state is q_i) from $rules(t(u_i))$. Note that, by definition, A' accepts t' with an accepting run $\overline{r}$. Since i is minimal and the run r' simulates r, the first $i-1$ rules applied in $\overline{r}$ are $\Pi_1, \ldots, \Pi_{i-1}$. Then, the left-state of the ith rule

of A' in $\overline{r}$ is also q_i. This means, by the definition of A', that there is a rule of A in $rules(t'(u_i))$ with left state q_i, i.e, there are more than one rules in $rules(t(u_i))$ with left-state q_i. This contradicts the definition of $\Sigma(P)$, hence t does not contain any superfluous rule.

To prove the other inclusion, assume that $t \in L(A')$ contains no superfluous rules. Let $s \in T_\Sigma$ be the tree obtained from t by dropping the rule sets from its labels. By the above discussion concerning A', we get that A accepts the tree s applying the rules in the nodes of t. Let r be the so obtained accepting run on s. Since all rules in the nodes of t are applied, t encodes r, hence $t \in Rtree_s \subseteq Rtree_A$. □

Next we informally introduce a 1-pebble twa A'' that accepts those trees in $L(A')$ which contain superfluous rules. Intuitively, the 1-ptwa A'' works as follows on an input tree $t \in T_{\Sigma(P)}$.

Phase 1: A'' nondeterministically chooses a node u of t and places the pebble at u. Assume that $t(u) = \langle \sigma, J \rangle$. If $J = \emptyset$ then, there is no next step and the computation terminates without acceptance.

Phase 2: If $J \neq \emptyset$, then A'' nondeterministically picks a rule $\pi = \langle q, \sigma, j \rangle \to \langle q', \varphi \rangle \in J$. Let $\Pi = \langle q, \langle \sigma, J \rangle, j \rangle \to \langle q', \varphi \rangle$ be the rule of A' corresponding to π. Our A'' stores Π in its state.

Phase 3: In the rest, A'' computes deterministically. First A'' moves back to the root node and then it simulates A' on t. However, during the simulation of A', our A'' is not allowed to use the rule Π (stored in its memory) at node u (being marked by the pebble).

It is clear that A'' has an accepting computation on t if and only if A' has an accepting computation on t which does not apply at least one rule in a label of t. Hence, we obtain the following result.

Lemma 14. $L(A'') = \{t \in T_{\Sigma(P)} \mid t \in L(A') \text{ and } t \text{ contains a superfluous rule}\}$.

Now we can prove the following statement easily.

Corollary 15. *The tree language $Rtree_A$ is effectively regular.*

Proof. It follows from Lemmata 13 and 14 that $Rtree_A = L(A') \cap \overline{L(A'')}$. Finally, by Propositions 1 and 7, we obtain that $Rtree_A$ is effectively regular. □

Now we are ready to prove our main result. For this we will need the relabeling $\tau : \Sigma(P) \to \Sigma$ which drops the rule component from each symbol, i.e., which is defined by $\tau(\langle \sigma, J \rangle) = \sigma$ for every $\langle \sigma, J \rangle \in \Sigma(P)$. Note, for later use, that $Rtree_s = \{t \in Rtree_A \mid \tau'(t) = s\}$.

Theorem 16. *The tree series S_A is effectively regular.*

Proof. By Corollary 15 and Proposition 3(a), the characteristic tree series $\mathcal{X}_{Rtree_A} : T_{\Sigma(P)} \to K$ is effectively regular. Moreover, since the tree series *weight* is effectively regular (see Lemma 8), we obtain by Proposition 3(b) that the Hadamard product $S = \mathcal{X}_{Rtree_A} \odot weight$ is an effectively regular tree series.

Let us note that for each tree $t \in T_{\Sigma(P)}$ we have

$$(S,t) = \begin{cases} (weight,t) & \text{if } t \in Rtree_A, \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, $\hat{\tau}(S)$ is a tree series in $K\langle\langle T_\Sigma \rangle\rangle$, were $\tau : \Sigma(P) \to \Sigma$ is the relabeling introduced above, and it follows from Proposition 4 and the fact that S is regular that $\hat{\tau}(S)$ is effectively regular. Finally, for every tree $s \in T_\Sigma$,

$$\begin{aligned} (\hat{\tau}(S), s) &= \sum_{t \in T_{\Sigma(P)},\, \tau'(t)=s} (S,t) = \sum_{t \in T_{\Sigma(P)},\, \tau'(t)=s} (\mathcal{X}_{Rtree_A} \odot weight, t) \\ &= \sum_{t \in Rtree_A,\, \tau'(t)=s} (weight,t) = \sum_{t \in Rtree_s} (weight,t) \\ &= (S_A, s), \end{aligned}$$

where the last equality is justified by Corollary 12. Hence $S_A = \hat{\tau}(S)$, which proves that S_A is also effectively regular. □

Since A is an arbitrary non-looping wtwa over K, we also proved the following result.

Theorem 17. $\mathrm{TWA}(\Sigma, K) \subseteq \mathrm{REG}(\Sigma, K)$.

In the remainder of the paper we show that the inclusion is strict provided K is proper. For this, let K' be another commutative semiring and $h : K \to K'$ a semiring homomorphism. Then we can prove easily the analogy of Proposition 5 for wtwa, i.e., that h preserves recognizability by wtwa.

Proposition 18. *(a) For every $S \in \mathrm{TWA}(\Sigma, K)$ and semiring homomorphism $h : K \to K'$, we have $h(S) \in \mathrm{TWA}(\Sigma, K')$, i.e., $h(\mathrm{TWA}(\Sigma, K)) \subseteq \mathrm{TWA}(\Sigma, K')$. (b) If, in addition, h is surjective, then $h(\mathrm{TWA}(\Sigma, K)) = \mathrm{TWA}(\Sigma, K')$.*

Proof. Let $A = (Q, \Sigma, q_0, q_a, P)$ be a wtwa over K. Construct the wtwa $A = (Q, \Sigma, q_0, q_a, P')$ such that $P' = \{\langle q, \sigma, j\rangle \xrightarrow{h(a)} \langle q', \varphi\rangle \mid \langle q, \sigma, j\rangle \xrightarrow{a} \langle q', \varphi\rangle \in P\}$. It is easy to see that $S_{A'} = h(S_A)$. Moreover, if h is surjective, then every wtwa over K' appears as the "image" of a wtwa over K. □

Now we recall an important result from [4], namely, that there is a regular tree language which cannot be recognized by any twa. It is well known that regular tree languages and regular tree series over $\mathbb{B}$ can be identified, cf. e.g. [18], and it is easy to see that the same holds for tree languages recognizable by twa as well as

for tree series recognizable by wtwa over $\mathbb{B}$. Thus, the above mentioned result of [4] can be written in the form $\mathrm{REG}(\Sigma, \mathbb{B}) - \mathrm{TWA}(\Sigma, \mathbb{B}) \neq \emptyset$. Then we can prove the following result and thus generalize the main result of [4] for tree series recognizable by wtwa over proper commutative semirings.

Theorem 19. *If K is proper, then* $\mathrm{REG}(\Sigma, K) - \mathrm{TWA}(\Sigma, K) \neq \emptyset$.

Proof. Assume, on the contrary, that $\mathrm{REG}(\Sigma, K) \subseteq \mathrm{TWA}(\Sigma, K)$. Since K is proper, by Theorem 2.1 of [27], there is a surjective homomorphism $h : K \to \mathbb{B}$. Then obviously we have $h(\mathrm{REG}(\Sigma, K)) \subseteq h(\mathrm{TWA}(\Sigma, K))$. On the other hand, by Propositions 5 and 18, we have $h(\mathrm{REG}(\Sigma, K)) = \mathrm{REG}(\Sigma, \mathbb{B})$ and $h(\mathrm{TWA}(\Sigma, K)) = \mathrm{TWA}(\Sigma, \mathbb{B})$, which is a contradiction. □

5 Conclusion and an open problem

We generalized tree-walking automata of [1] by equipping each transition rule with a weight taken from a semiring K. For two reasons, we considered the non-looping model, i.e., which cannot fall into infinite computation from the initial configuration. The first reason is that the weight of an input tree s is defined as the sum of the weights of the accepting runs on s. The second one is that the proofs of Corollaries 12 and 15, and hence of Theorem 16 work only for non-looping wtwa.

If a wtwa is looping, then there may be infinitely many accepting runs on s, hence computing the weight of s leads to an infinite sum in the semiring. This problem can be handled by considering underlying semirings which are *complete*, i.e., in which the sum of infinitely many elements exists [21, 22]. Therefore, it is an open problem whether our main result can be generalized to arbitrary (including looping) wtwa over complete semirings.

Acknowledgment

We are grateful to the anonymous referees for their effort and valuable comments.

References

[1] A. V. Aho and J. D. Ullman. Translations on a context-free grammar. *Inform. Control*, 19:439–475, 1971.

[2] A. Alexandrakis and S. Bozapalidis. Weighted grammars and Kleene's theorem. *Information Processing Letters*, 24(1):1–4, January 1987.

[3] J. Berstel and C. Reutenauer. Recognizable power series on trees. *Theoret. Comput. Sci.*, 18:115–148, 1982.

[4] M. Bojańczyk and T. Colcombet. Tree-walking automata do not recognize all regular languages. *SIAM J. Comput.*, 38(3):658–701, 2008.

[5] B. Borchardt. The Myhill-Nerode Theorem for Recognizable Tree Series. In *7th International Conference on Developments in Language Theory, DLT'03, Szeged, Hungary, July 7-11, 2003, Proceedings*, volume 2710 of *LNCS*, pages 146–158. Springer-Verlag, July 2003.

[6] B. Borchardt. A Pumping Lemma and Decidability Problems for Recognizable Tree Series. *Acta Cybernet.*, 16(4):509–544, 2004.

[7] B. Borchardt, A. Maletti, B. Šešelja, A. Tepavčevic, and H. Vogler. Cut sets as recognizable tree languages. *Fuzzy Sets and Systems*, 157:1560–1571, 2006.

[8] B. Borchardt and H. Vogler. Determinization of Finite State Weighted Tree Automata. *Journal of Automata, Languages and Combinatorics*, 8(3):417–463, 2003.

[9] H. Comon, M. Dauchet, R. Gilleron, F. Jacquema, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.

[10] J. Doner. Tree acceptors and some of their applications. *J. Comput. System Sci.*, 4:406–451, 1970.

[11] M. Droste, C. Pech, and H. Vogler. A Kleene theorem for weighted tree automata. *Theory of Computing Systems*, 38:1–38, 2005.

[12] M. Droste and H. Vogler. Weighted tree automata and weighted logics. *Theoret. Comput. Sci.*, 366:228–247, 2006.

[13] J. Engelfriet and H. J. Hoogeboom. Tree-walking pebble automata. In *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 72–83, London, UK, 1999. Springer-Verlag.

[14] J. Engelfriet and H. J. Hoogeboom. Automata with nested pebbles capture first-order logic with transitive closure. *Logical Methods in Computer Science* 3(2007), Issue 2, Paper 3.

[15] J. Engelfriet, H. J. Hoogeboom, and B. Samwel. XML transformation by tree-walking transducers with invisible pebbles. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '07)*, pages 63–72, New York, NY, USA, 2007. ACM Press.

[16] Z. Ésik and W. Kuich. Formal Tree Series. *Journal of Automata, Languages and Combinatorics*, 8:219–285, 2003.

[17] Z. Fülöp and L. Muzamel. Circularity and Decomposition Results for Pebble Macro Tree Transducers. *Journal of Automata, Languages and Combinatorics*, 13(1):3–44, 2008.

[18] Z. Fülöp and H. Vogler. Weighted tree automata and tree transducers. In M. Droste, W. Kuich, and H. Vogler, editors, *Handbook of Weighted Automata*, Chapter 9. Springer-Verlag, 2009.

[19] F. Gécseg and M. Steinby. *Tree Automata.* Akadémiai Kiadó, Budapest, 1984.

[20] F. Gécseg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 1–68. Springer-Verlag, 1997.

[21] U. Hebisch and H. J. Weinert. *Semirings - Algebraic Theory and Applications in Computer Science.* World Scientific, Singapore, 1998.

[22] W. Kuich. Formal power series over trees. In S. Bozapalidis, editor, *3rd International Conference on Developments in Language Th eory, DLT 1997, Thessaloniki, Greece, Proceedings*, pages 61–101. Aristotle University of Thessaloniki, 1998.

[23] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. *J. of Comput. Syst. Sci.*, 66:66–97, 2003.

[24] L. Muzamel. Pebble Alternating Tree-Walking Automata and Their Recognizing Power. *Acta Cybernetica*, 18(3):427–450, 2008.

[25] F. Neven. Automata theory for XML researchers. *SIGMOD Rec.*, 31(3):39–46, 2002.

[26] T. Schwentick. Automata for XML – A survey. *J. Comput. Syst. Sci.*, 73(3):289–315, 2007.

[27] H. Wang. On characters of semirings. *Houston J. Math.*, 23:391–405, 1997.

Received 11th July 2008

Acta Cybernetica **19** (2009) 295–312.

Weighted Automata Define a Hierarchy of Terminating String Rewriting Systems

Andreas Gebhardt* and Johannes Waldmann†

Abstract

The "matrix method" (Hofbauer and Waldmann 2006) proves termination of string rewriting via linear monotone interpretation into the domain of vectors over suitable semirings. Equivalently, such an interpretation is given by a weighted finite automaton. This is a general method that has as parameters the choice of the semiring and the dimension of the matrices (equivalently, the number of states of the automaton). We consider the semirings of non-negative integers, rationals, algebraic numbers, and reals; with the standard operations and ordering. Monotone interpretations also allow to prove relative termination, which can be used for termination proofs that consist of several steps. The number of steps gives another hierarchy parameter. We formally define the hierarchy and we prove that it is infinite in both directions (dimension and steps).

Keywords: string rewriting, relative termination, weighted automaton, matrix interpretation, monotone algebra.

1 Introduction

Rewriting is pattern replacement in context. It serves as a model of computation that is Turing-complete. Thus all "interesting" semantic properties are undecidable, including the very natural question of *termination* [18]: for a given rewriting system, are all derivations finite? Since the problem is significant in practice, e.g. for the analysis of software, one is interested in semi-algorithms: computable methods of proving termination that are sound, but not complete.

One method to prove termination of rewriting is "matrix interpretation" [13]. These interpretations are in fact $\mathbb{N}$-weighted finite automata. Several automated termination provers now implement this method, and indeed the outcome of recent Termination Competitions is heavily influenced by "matrix proofs".

*Martin-Luther-Universität Halle-Wittenberg, Von-Seckendorff-Platz 1, D-06120 Halle , Germany. E-mail: andreas.gebhardt@informatik.uni-halle.de

†Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fb IMN, PF 30 11 66, D-04251 Leipzig, Germany. E-mail: waldmann@imn:htwk-leipzig.de

Related to that, investigations of matrix method(s) mainly focused on proving correctness, and then efficiency of implementation in solving the corresponding constraint systems for the matrix entries.

With the present paper, we intend to start a systematic study of matrix method(s) as *proof systems*. We define a suitable hierarchy of termination problems and explore its properties.

One parameter of this hierarchy is the size of the matrices used in the proof, corresponding to the number of states of the automata.

Another parameter is the underlying (semi)ring. In the present paper, we consider weight rings that include $\mathbb{N}$. In [8] we reported on some experiments with non-negative rationals.

The matrix interpretation method in fact solves a more general problem: that of *relative termination*. A rewriting system R terminates relative to a rewriting system S if each mixed derivation (containing R and S steps in any order) contains only finitely many R steps. While being an interesting concept in itself [10], relative termination helps to solve standard termination problems because it allows to compose termination proofs: if R terminates relative to S then termination of $R \cup S$ follows from termination of S, and the latter can be proved separately. That way, termination of a rewriting system can be shown incrementally, and the number of proof steps gives another interesting parameter for the hierarchy.

In the present paper, we focus on string rewriting. The matrix method has been generalized to term rewriting [6], but we leave the investigation of the corresponding hierarchy of terminating term rewriting systems for further study.

After giving preliminaries on string rewriting in Section 2 and on termination proofs via weighted word automata in Section 3, we define the corresponding hierarchy of (relatively) terminating rewriting systems in Section 4. Then we discuss the hierarchy with respect to matrix dimension in Section 5 (with a particular case in Section 6), choice of the weight semiring in Section 7, and number of proof steps in Section 8.

We obtain these results:

- the hierarchy is infinite with respect to matrix dimension (Theorem 2)
- rational weights are strictly more powerful than integral weights (Theorem 5)
- the hierarchy is infinite with respect to the number of proof steps (Theorem 6).

Some of the results in this paper have been announced in contributions to the Workshop on Termination [8] and to the Workshop on Weighted Automata [9].

2 Notation and Preliminaries

Strings and Rewriting. Given a finite alphabet Σ, denote by Σ^* the set of finite words with letters from Σ. In fact Σ^* is a monoid under the operation $\cdot$ of concatenation, with the empty word ϵ as unit.

A *string rewriting system* [3] is a set R of rules, where a rule is a pair of words. We often write the rule (l, r) as $(l \to r)$. A string rewriting system R defines a (one-step) rewrite relation over Σ^* by $u \to_R v$ if there exists $(l, r) \in R$ and $x, y \in \Sigma^*$ such that $u = x \cdot l \cdot y$ and $v = x \cdot r \cdot y$. For example, for $R = \{ab \to baa\}$ over $\Sigma = \{a, b\}$, we have $abb \to_R baab \to_R babaa \to_R bbaaaa$. We often write R (the system) as a shorthand for $\to_R$ (the relation).

Relations and Termination. For a relation $\to$, we write $\mathrm{SN}(\to)$ if this $\to$ is well-founded, that is, if there is no infinite chain $x_0 \to x_1 \to \ldots$ We also say that $\to$ is *terminating*.

We denote the composition of relations $\to_1$ and $\to_2$ by $\to_1 \circ \to_2$, the transitive closure of a relation $\to$ by $\to^+$, and the transitive and reflexive closure by $\to^*$.

For relations $\to_1, \to_2$, define $\to_1 \,/\, \to_2$ as $\to_1 \circ \to_2^*$. Then $\mathrm{SN}(\to_1 \,/\, \to_2)$ denotes that $\to_1$ is *terminating relative to* $\to_2$: there is no $(\to_1 \cup \to_2)$-chain containing infinitely many $\to_1$ steps. Note that $\to_1 /\emptyset = \to_1$.

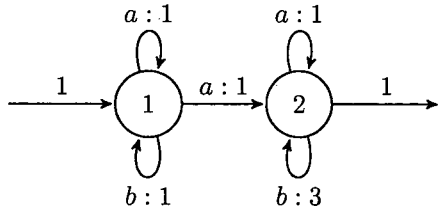
By the above remark, we write $\mathrm{SN}(R)$ ("the system R is terminating") for $\mathrm{SN}(\to_R)$ ("the derivation relation of R is terminating").

Semirings. A *semiring* [11] has a carrier D with operations $+$ (addition) and $\cdot$ (multiplication) and designated elements 0 (zero) and 1 (unit), such that $(D, +, 0)$ is a commutative monoid, and $(D, \cdot, 1)$ is a monoid, addition distributes over multiplication from both sides, and $0 \cdot a = 0 = a \cdot 0$. A semiring is *partially ordered* [7] if there is a relation $\geq$ on D that is compatible with the operations. In the present paper, we use semirings over the domains of natural numbers $\mathbb{N}$, non-negative rational numbers $\mathbb{Q}_{\geq 0}$, algebraic numbers $\mathrm{Alg}_{\geq 0}$, and real numbers $\mathbb{R}_{\geq 0}$; each with standard operations. For $\mathbb{N}$, we use the standard ordering; for the others, see below (after Theorem 1). The given domains are in fact positive cones of rings, but we rarely subtraction.

Weighted automata. A *weighted automaton* [2, 5, 15] $A = (D, \Sigma, Q, \lambda, \mu, \gamma)$ consists of a semiring D, an alphabet Σ, a set of states Q, and mappings

$$\lambda : Q \to D, \mu : (Q \times \Sigma \times Q) \to D, \gamma : Q \to D.$$

We picture such an automaton as a directed labelled graph (possibly with loops and parallel edges), with an edge $p \xrightarrow{x:w} q$ for each $\mu(p, x, q) = w$. An incoming edge (no source) $\xrightarrow{w} q$ denotes $\lambda(q) = w$, an outgoing edge (no target) $p \xrightarrow{w}$ denotes $\gamma(p) = w$. We omit all edges with weight 0. As an ongoing example for this section,



A *path* in the automaton is a sequence $q_0 \xrightarrow{x_1:w_1} q_1 \xrightarrow{x_2:w_2} \cdots \xrightarrow{x_n:w_n} q_n$. The *label* of this path is $x_1 x_2 \ldots x_n \in \Sigma^*$, and the *weight* of this path is $w_1 \cdot w_2 \cdot \cdots \cdot w_n \in D$. For instance, the path $1 \xrightarrow{a:1} 1 \xrightarrow{a:1} 2 \xrightarrow{b:3} 2$ has label aab and weight 3. For each state q, there is an empty path from q to q with label ϵ and weight 1.

The function $\mu^* : (Q \times \Sigma^* \times Q) \to D$ computes the weight of a word $x = x_1 x_2 \ldots x_n$ from state q_0 to q_n as the sum of the weights of all paths from p to q with label x:

$$\mu^*(q_0, x_1 \ldots x_n, q_n) = \sum_{q_1,\ldots,q_{n-1} \in Q} \prod_{1 \le k \le n} \mu(q_{k-1}, x_k, q_k)$$

For instance, $\mu^*(1, aab, 2)$ is computed from the paths $1 \xrightarrow{a:1} 1 \xrightarrow{a:1} 2 \xrightarrow{b:3} 2$ and $1 \xrightarrow{a:1} 2 \xrightarrow{a:1} 2 \xrightarrow{b:3} 2$, so the total weight is 6. We identify μ^* with μ, and find it convenient to write $\mu(p, x, q) = d$ as $p \xrightarrow{x:d}_A q$.

The weight assigned by A to a word w is obtained by considering the functions λ and γ that give the weights for entering and leaving a state,

$$A(w) = \sum_{i,f \in Q} \lambda(i) \cdot \mu^*(i, w, f) \cdot \gamma(f).$$

In the example, $A(aab) = A(1, aab, 2) = 6$.

We say that state $q \in Q$ is *initial* if $\lambda(q) = 1$, and zero elsewhere; and q is *final* if $\gamma(q) = 1$, and zero elsewhere. An automaton with unique initial state i and unique final state f is called (i, f)-*pointed*.

Reduced automata. We say that states p *is connected to* state q in A if there is some $w \in \Sigma^*$ such that $\mu(p, w, q) \neq 0$. We write $p \to_A^* q$. An (i, f)-pointed automaton is called *reduced* if for each $q \in Q$, we have $i \to_A^* q \to_A^* f$. For each automaton A, there is a reduced automaton A' that computes the same weight function as A. This A' can be obtained from A by simply deleting all unconnected states.

Matrices. The function μ of a weighted automaton can also be visualized as a mapping that assigns to each letter $x \in \Sigma$ a square matrix, also called $\mu(x)$, that is indexed by $Q \times Q$. For the example automaton, we have these matrices

$$\mu(a) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \mu(b) = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$$

This mapping can be extended from letters to words, by matrix multiplication: $\mu(x_1 \ldots x_n) = \mu(x_1) \cdot \ldots \cdot \mu(x_n)$, and this corresponds with the function μ^* defined above, that is, the entry at position (p, q) in the matrix product $\mu(x_1 \ldots x_n)$ is the weight of the word $x_1 \ldots x_n$ from p to q, as defined above. In the example, we compute

$$\mu(aab) = \mu(a) \cdot \mu(a) \cdot \mu(b) = \begin{pmatrix} 1 & 6 \\ 0 & 3 \end{pmatrix}.$$

If we view λ as a row vector and γ as a column vector, then $A(w) = \lambda \cdot \mu(w) \cdot \gamma$. For example,

$$A(aab) = \lambda \cdot \mu(aab) \cdot \gamma = \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 6 \\ 0 & 3 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 6$$

For (i, f)-pointed automata, λ and γ are unit vectors, so $A(w)$ is just the entry at position (i, f) in the square matrix $\mu(w)$. Usually, i is the first index and f is last, so (i, f) marks the top right position.

3 Termination Proofs from Weighted Automata

An (i, f)-pointed automaton A is is called *weakly compatible with a rewriting system* R, if $\forall a \in \Sigma : \mu(i, a, i) \geq 1 \wedge \mu(f, a, f) \geq 1$ and for each rule $(l \to r) \in R$, and states $p, q \in Q$, we have $\mu(p, l, q) \geq \mu(p, r, q)$. The automaton is called *strictly compatible* with R if additionally for each rule $(l \to r) \in R$, $\mu(i, l, f) > \mu(i, r, f)$. (In this paper we only use sub-semirings of $\mathbb{R}_{\geq 0}$, so all weights are non-negative.)

The main result of [13], written here in the language of weighted automata, is

Theorem 1. *If there is an $\mathbb{N}$-weighted automaton A that is strictly compatible with a rewriting system R and weakly compatible with a rewriting system S, then* $\mathrm{SN}(R/S)$. □

The intution is that for a rewrite step $xly \to xry$ using a rule $(l \to r) \in R$, each path $i \xrightarrow{x:*} i \xrightarrow{l:*} f \xrightarrow{y:*} f$ has strictly larger weight than the corresponding path $i \xrightarrow{x:*} i \xrightarrow{r:*} f \xrightarrow{y:*} f$. The total weight of xly (xry, resp.) may include contributions from other paths, but for these we require a weak decrease. By strictness of "$<$" w.r.t. addition, we get a total decrease.

We give an example where Theorem 1 is applied with $S = \emptyset$.

Example 1. For the rewriting system $R = \{ab \to baa\}$, consider the $(1, 2)$-pointed automaton with transition matrices

$$\mu(a) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \mu(b) = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$$

By matrix multiplication, we compute

$$\mu(ab) = \begin{pmatrix} 1 & 3 \\ 0 & 3 \end{pmatrix}, \quad \mu(baa) = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}$$

and we note $\mu(i, ab, f) = 3 > 2 = \mu(i, baa, f)$, and weak inequalities (in fact, equalities) elsewhere. This shows that the automaton is strictly compatible with R. From the theorem, we conclude $\mathrm{SN}(R/\emptyset)$, thus $\mathrm{SN}(R)$. □

In [8] it was observed that the theorem also holds if we replace $\mathbb{N}$ by $\mathbb{Q}_{\geq 0}$, and this easily extends to $\mathrm{Alg}_{\geq 0}$ and $\mathbb{R}_{\geq 0}$. Now $>$ is not well-founded on $\mathbb{Q}_{\geq 0}$ and indeed we use a different ordering: $x >_\epsilon y \iff x \geq \epsilon + y$ where

$$\epsilon = \inf \{\mu(i, l, f) - \mu(i, r, f) \mid (l \to r) \in R\}.$$

If the automaton is strictly compatible with a finite system R, then this is a positive number, and therefore $>_\epsilon$ is well-founded. Under the conditions of the theorem, we have $u \to_R v$ implies $\mu(i,u,f) >_\epsilon \mu(i,v,f)$.

The following is an easy observation:

Lemma 1. *If A fulfills the conditions of Theorem 1, then there is a reduced automaton A' with the same properties.*

Proof. We take A' as the reduced automaton of A, obtained by deleting states that are unreachable from i or do not reach f. Denote by μ' the transition function of A'. For states p, q of A', and letter $x \in \Sigma$, we have $\mu'(p,x,q) = \mu(p,x,q)$. Therefore, also for $w \in \Sigma^*$ we have $\mu'(p,w,q) = \mu(p,w,q)$. Since initial and final state of A and A' coincide (respectively), we are done. □

The following example shows an application of the theorem with non-empty S.

Example 2. Take $S = \{ab \to baa\}$ and $R = \{cb \to bcc\}$, and the $(1,2)$-pointed automaton with matrices

$$\mu(a) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mu(b) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \mu(c) = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}$$

We compute

$$\mu(cb) = \begin{pmatrix} 3 & 3 \\ 0 & 1 \end{pmatrix}, \quad \mu(bbc) = \begin{pmatrix} 3 & 2 \\ 0 & 1 \end{pmatrix}, \quad \mu(ab) = \mu(baa) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

This shows that the automaton is strictly compatible with R and weakly compatible with S, thus $\mathrm{SN}(R/S)$. □

Now we introduce an additional notation:

Definition 1. *For rewriting systems R, S we write*

$$R \vdash S \quad \Longleftrightarrow \quad R \supseteq S \wedge \mathrm{SN}((R \setminus S)/S).$$

Example 3. Example 1 shows that $\{ab \to baa\} \vdash \emptyset$. Example 2 shows that $\{ab \to baa, cb \to bbc\} \vdash \{ab \to baa\}$. □

Proposition 1. *$R \vdash S$ if and only if each infinite R-derivation ends with an infinite S-derivation.*

Proof. We have $\mathrm{SN}((R \setminus S)/S)$ if and only if each R-derivation contains only finitely many steps from $R \setminus S$. □

The notation "$\vdash$" supports the idea of composing termination proofs. Indeed,

Proposition 2. *The relation $\vdash$ is transitive.*

Proof. Given $R \vdash S$ and $S \vdash T$, we have to show that each infinite R-derivation ends with an infinite T-derivation. Assume there is an R-derivation with infinitely many steps from $R \setminus T$. If this derivation contains infinitely many steps from $R \setminus S$, then this contradicts $R \vdash S$. So it contains only finitely many steps from $R \setminus S$. After the last of these, we have an S-derivation. By $S \vdash T$, it contains only finitely many steps from $S \setminus T$, and then continues as an infinite T-derivation. □

We obtain the following

Corollary 1. *If* $R \vdash^* \emptyset$, *then* $\mathrm{SN}(R)$. □

Here is a typical application:

Example 4. By Example 3, we have

$$\{ab \to baa, cb \to bbc\} \vdash \{ab \to baa\} \vdash \emptyset,$$

thus the rewriting system on the left is terminating. □

4 A Hierarchy of Relative Termination

We relate the general idea of relative termination, as denoted by "$\vdash$", with the idea of matrix interpretations.

Definition 2. *We denote by* $\mathfrak{M}(W, n)$ *the set of pairs of rewriting systems* (R, S) *for which an automaton exists with weight domain* W *and* n *states that is strictly compatible with* $R \setminus S$ *and weakly compatible with* S. *We also write* $R \vdash^{\mathfrak{M}(W,n)} S$.

Indeed $\mathfrak{M}(W, n)$ is a relation on rewriting systems, and by Theorem 1, we have that $R \vdash^{\mathfrak{M}(W,n)} S$ implies $R \vdash S$.

For relations $\mathfrak{M}(W, n)$ we will make use of standard operations on relations like composition, iteration (exponentiation) and (reflexive and) transitive closure.

Example 5. By Example 1 we get $\{ab \to baa\} \vdash^{\mathfrak{M}(\mathbb{N},2)} \emptyset$. □

By abuse of notation we sometimes write $R \in \mathfrak{M}(W, d)$ for $(R, \emptyset) \in \mathfrak{M}(W, d)$.

Example 6. By Examples 1,2, and the above abuse of notation,

$$\{ab \to baa, cb \to bbc\} \in \mathfrak{M}(\mathbb{N}, 2)^2.$$

The exponent 2 indicates that the termination proof is composed of two steps. □

Definition 3. *The* matrix termination hierarchy *consists of the classes* $\mathfrak{M}(W, d)^s$ *of pairs of rewriting systems, where*

- $W \in \{\mathbb{N}, \mathbb{Q}_{\geq 0}, \mathrm{Alg}_{\geq 0}, \mathbb{R}_{\geq 0}\}$ *is a weight semiring,*

- *d is a natural number ≥ 0 giving the matrix dimension (automaton size),*
- *and s is a natural number ≥ 1 counting the proof steps s.*

We abbreviate $\cup_{n \geq 0} \mathfrak{M}(W, n)$ by $\mathfrak{M}(W)$. Then in our notation $\mathfrak{M}(\mathbb{N})$ is the set of all rewriting systems that have a one-step termination proof using some natural-weighted automaton. Using transitivity, $\mathfrak{M}(\mathbb{N})^+$ is the set of all systems with a multi-step termination proof using such automata.

We have these immediate observations:

Proposition 3. *1. If $n \leq n'$, then for all W, $\mathfrak{M}(W, n) \subseteq \mathfrak{M}(W, n')$.*

2. If W is a sub-semiring of W', then for all n, $\mathfrak{M}(W, n) \subseteq \mathfrak{M}(W', n)$.

3. If $1 \leq s \leq s'$, then for all W, n

$$\mathfrak{M}(W, n) \subseteq \mathfrak{M}(W, n)^{\leq s} \subseteq \mathfrak{M}(W, n)^{\leq s'} \subseteq \mathfrak{M}(W, n)^*.$$

Proof. (1) We can introduce useless states in the automaton. (2) Each W-interpretation is also a W'-interpretation. (3) Each sequence with $\leq s$ steps is also a sequence with $\leq s'$ steps. □

While these statements are obvious, the following problems are not:

- Which of the obvious inclusions are strict?
- Are there non-obvious inclusions?
- Are the hierarchies (w.r.t. number of states, number of steps) infinite?
- What levels $\mathfrak{M}(W, n)^s$ are inhabited?

We will answer some of them in the rest of the paper.

5 Number of States

In this section we present a terminating rewriting system that needs large matrices for a termination proof. The construction works for any size, so we infer that the "matrix size hierarchy" is infinite.

We consider, for $d \geq 2$, the alphabet $\Sigma_d = \{s, 1, \ldots, d, f\}$. These are d numbers and two extra letters s, f (start and final). We take any enumeration $e_1, \ldots$ of even permutations of $\{1, \ldots, d\}$ and enumeration $o_1, \ldots$ of odd permuations of $\{1, \ldots, d\}$. Then consider the string rewriting system

$$R_d = \{se_k f \to so_k f \mid 1 \leq k \leq d!/2\}.$$

Example 7. For $d = 4$, we get the rule set

$$\begin{array}{lll} s1234f \to s2134f, & s2314f \to s2341f, & s3124f \to s1324f, \\ s3241f \to s3214f, & s1342f \to s3142f, & s3412f \to s3421f, \\ s2143f \to s1243f, & s2431f \to s2413f, & s1423f \to s4123f, \\ s4213f \to s4231f, & s4132f \to s1432f, & s4321f \to s4312f. \end{array}$$

Lemma 2. *There is no strict subset S of R_{2d} such that $(R_{2d}, S) \in \mathfrak{M}(\mathbb{N}, d)$.*

Proof. We use the Amitsur-Levitzki Theorem [14, 4]. It says that the elementary symmetric polynomial in $2d$ variables

$$s(x_1, \ldots, x_{2d}) = \sum_{\pi \text{ is a permutation of } \{1,\ldots,2d\}} (-1)^{\mathrm{sgn}(\pi)} x_{\pi(1)} \cdot \ldots \cdot x_{\pi(2d)}$$

is identically zero for $d \times d$-matrices.

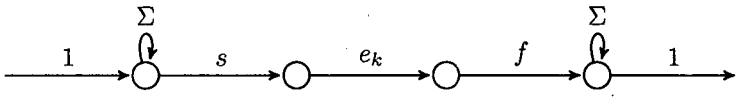
Any d-dimensional matrix interpretation $[\cdot]$ has

$$\sum_{(l \to r) \in R} ([l] - [r]) = [s] \left(\sum ([e_k] - [o_k]) \right) [f] = 0.$$

If $[\cdot]$ is weakly compatible with R_{2d}, then $\sum_{(l \to r) \in R_{2d}} ([l] - [r]) \geq 0$, and this implies $\forall (l \to r) \in R_{2d} : [l] = [r]$. So, $[\cdot]$ cannot be strictly compatible with any rule of R_{2d}. □

Lemma 3. *For $d' = 2d + 3$, $R_{2d} \in \mathfrak{M}(\mathbb{N}, d')^{(2d)!/2}$*

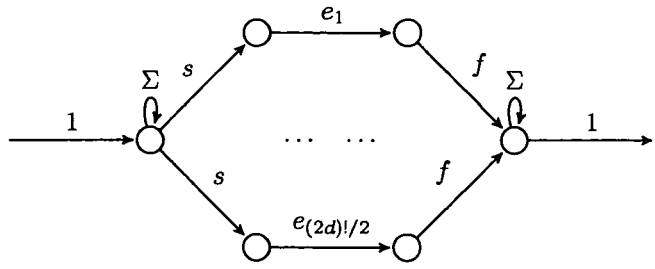
Proof. For each k, we give a matrix interpretation $[\cdot]$ of dimension d' that is weakly compatible with all rules of R_{2d} and strictly compatible with rule $se_kf \to so_kf$. The interpretation represents an automaton that just counts the number of factors se_kf. This is a word with $2d + 2$ letters, so counting can be done with $2d + 3$ states. The counting automaton consists of loops at initial and final state, and a path labelled se_kf (and all unit weights) from initial to final state.



This works since se_kf is not self-overlapping (no non-trivial prefix is equal to a suffix). The count reduces by one at each rewrite step, since there are no overlaps between se_kf and $so_{k'}f$ either. Applying these interpretations for all k, in any order, gives the result: termination of R_{2d} can be shown by a sequence of $(2d)!/2$ matrix interpretations of size d'. □

Lemma 4. *For $d' = 2 + (2d + 1)(2d)!/2$, we have $R_{2d} \in \mathfrak{M}(\mathbb{N}, d')$.*

Proof. We build an automaton that contains all the automata constructed in the proof of Lemma 3 in parallel.



It has one initial and one final state, and $(2d)!/2$ paths each using $(2d+1)$ individual states. □

As a corollary, we obtain

Theorem 2. *For each $W \in \{\mathbb{N}, \mathbb{Q}_{\geq 0}, \mathrm{Alg}_{\geq 0}, \mathbb{R}_{\geq 0}\}$: The hierarchy $\mathfrak{M}(W,d)_{d=0,1,\ldots}$ is infinite.*

Proof. Assume, to the contrary, that there is d such that $\mathfrak{M}(W,d) = \mathfrak{M}(W,d+1) = \ldots$ By Lemma 2, the system R_{2d} is not in $\mathfrak{M}(W,d)$, and by Lemma 4, $R_{2d} \in \mathfrak{M}(W,d')$ for some $d' > d$. □

6 Small Automata

We have more information on the lower levels of the hierarchy:

Proposition 4. *These inclusions are strict:*

$$\mathfrak{M}(\mathbb{N},0) \subset \mathfrak{M}(\mathbb{N},1) \subset \mathfrak{M}(\mathbb{N},2) \subset \mathfrak{M}(\mathbb{N},3).$$

Proof. We prove $R_1 = \{a \to b\} \in \mathfrak{M}(\mathbb{N},1) \setminus \mathfrak{M}(\mathbb{R}_{\geq 0},0)$. A strictly compatible 1-dimensional interpretation of the required shape is given by $[a] = 2, [b] = 1$. Any interpretation in $\mathfrak{M}(\mathbb{N},0)$ is necessarily constant, so it is strictly compatible only with the empty set of rules, and not with R_1.

We prove $R_2 = \{ab \to ba\} \in \mathfrak{M}(\mathbb{N},2) \setminus \mathfrak{M}(\mathbb{R}_{\geq 0},1)$. A strictly compatible 2-dimensional interpretation is given by

$$[a] = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, [b] = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Any one-dimensional matrix interpretation $[\cdot]$ is commutative, so $[ab] = [ba]$ and it cannot be strictly compatible with R_2.

We prove $R_3 = \{aa \to aba\} \in \mathfrak{M}(\mathbb{N},3) \setminus \mathfrak{M}(\mathbb{R}_{\geq 0},2)$. A strictly compatible 3-dimensional interpretation is

$$[a] = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}, [b] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Any two-dimensional interpretation $[\cdot]$ of the required shape has main diagonal entries ≥ 1 and thus $[aba] \geq [aa]$, contradicting strict compatibility with R_3. □

If the automata under consideration have only one state, the weight domain is not really important, and the "step hierarchy" collapses.

Lemma 5. $\mathfrak{M}(\mathbb{R}_{\geq 0}, 1) \subseteq \mathfrak{M}(\mathbb{N}, 1)$.

Proof. An interpretation $[\cdot]$ by a one-state $\mathbb{R}_{\geq 0}$-weighted automaton corresponds to a multiplicative weight assignment (the weight of a word is the product of the weight of its letters). Note that all weights are positive, by definition. Taking logarithms, we get an additive assignment (the weight of a word is the sum of its letter weights). The conditions of weak and strict compatibility give rise to a system of linear equalities and inequalities between letter weights. The coefficients are natural numbers (namely, numbers of occurences of letters in sides of rules). If such a system has any solution at all, then it also has a rational solution. Since the system is moreover homogenous (the linear functions contain no absolute parts), any rational solution can be scaled to give an integer solution. In fact the components are naturals, since weights must be non-negative. From natural additive weights we can get back to multiplicative weights by exponentiation. If we take any natural base, then the weights are natural (they are powers of the base). □

Example 8. For $R = \{aaa \to bca\}, S = \{b \to cac\}$ we obtain the system of inequalities

$$\log[a] \geq 0 \wedge \log[b] \geq 0 \wedge \log[c] \geq 0$$
$$\wedge 2\log[a] - \log[b] - \log[c] > 0 \wedge -\log[a] + \log[b] - 2\log[c] \geq 0.$$

One solution is $\log[a] = 4, \log[b] = 6, \log[c] = 1$. We can take base 2 and obtain multiplicative weights $[a] = 16, [b] = 64, [c] = 2$. This proves $R \cup S \vdash^{\mathfrak{M}(\mathbb{N},1)} S$. □

As a corollary to Lemma 5, we obtain

Theorem 3. $\mathfrak{M}(\mathbb{N}, 1) = \mathfrak{M}(\mathbb{Q}_{\geq 0}, 1) = \mathfrak{M}(\mathrm{Alg}_{\geq 0}, 1) = \mathfrak{M}(\mathbb{R}_{\geq 0}, 1)$. □

Now we consider the number of proof steps when using one-state automata. We show that two-step proofs are not stronger than one-step proofs.

Lemma 6. $\mathfrak{M}(\mathbb{N}, 1)^2 = \mathfrak{M}(\mathbb{N}, 1)$

Proof. We are given a two-step one-dimensional termination proof, and we need to construct an equivalent one-step proof. Assume weight function f is strictly compatible with R and weakly compatible with $S \cup T$, and weight function g is strictly compatible with S and weakly compatible with T. We construct a weight function h that is strictly compatible with $R \cup S$ and weakly compatible with T, as follows. (In light of the previous, we write the weight functions additively.)

We will define

$$h(x) = f(x) \cdot c + g(x),$$

for a suitable natural number $c > 0$. Such an interpretation h is weakly compatible with T, since both f and g have this property. Interpretation h is strictly compatible with S, since f is weakly compatible with S and g is strictly compatible with S.

We put

$$c = 1 + \sup\{\max(0, g(r) - g(l)) \mid (l \to r) \in R\}$$

This is one plus the maximal increase of g weights, for R rules.

It remains to check that h is strictly compatible with R. If $u \to_R v$, then $f(u) - f(v) \geq 1$ by strict compatibility (and using that weights are natural), and $g(u) - g(v) \geq -c + 1$ by definition of c. By definition of h we get $h(u) - h(v) \geq c + (-c + 1) = 1$, and this proves the claim. □

Example 9. We have $\{a^2 \to b^3, b^5 \to a^3\} \vdash^{\mathfrak{M}(\mathbb{N},1)} \{b^5 \to a^3\}$ by the interpretation $f : a \mapsto 5, b \mapsto 3$; and $\{b^5 \to a^3\} \vdash^{\mathfrak{M}(\mathbb{N},1)} \emptyset$ by $g : a \mapsto 0, b \mapsto 1$. Since $g(a^2) = 0, g(b^3) = 3$, we put $c = 4$ and get $\{a^2 \to b^3, b^5 \to a^3\} \vdash^{\mathfrak{M}(\mathbb{N},1)} \emptyset$ by $h : a \mapsto 20, b \mapsto 13$. □

As a corollary to Lemma 6, we obtain

Theorem 4. $\mathfrak{M}(\mathbb{N}, 1)^* = \mathfrak{M}(\mathbb{N}, 1)$ □

7 Choice of Weight Domain

We compare the power of matrix interpretations w.r.t. the weight domain.

We give an example $(R \cup S, S) \in \mathfrak{M}(\mathbb{Q}_{\geq 0}, 3)^2 \setminus \mathfrak{M}(\mathbb{N})^*$, that is, with a two-step termination proof of rational-weighted automata of size 3, but no natural-weighted termination proof of any size and number of steps.

The rewriting systems are

$$R = \{baa \to abc, ca \to ac, cb \to ba\}, S = \{\epsilon \to b\}.$$

Lemma 7. $(R \cup S, S) \in \mathfrak{M}(\mathbb{Q}_{\geq 0}, 3) \circ \mathfrak{M}(\mathbb{N}, 1)$.

Proof. We use the following interpretation

$$[a] = \begin{pmatrix} 1 & 1 & 0 \\ 0 & \frac{5}{2} & 6 \\ 0 & 0 & 1 \end{pmatrix}, [b] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}, [c] = \begin{pmatrix} 1 & 2 & 0 \\ 0 & \frac{5}{2} & \frac{7}{2} \\ 0 & 0 & 1 \end{pmatrix}.$$

giving these interpretations for the rules:

$$[baa] = \begin{pmatrix} 1 & \frac{7}{2} & 6 \\ 0 & \frac{25}{8} & \frac{21}{2} \\ 0 & 0 & 1 \end{pmatrix} \quad [abc] = \begin{pmatrix} 1 & \frac{13}{4} & \frac{7}{4} \\ 0 & \frac{25}{8} & \frac{83}{8} \\ 0 & 0 & 1 \end{pmatrix}$$

$$[ca] = \begin{pmatrix} 1 & 6 & 12 \\ 0 & \frac{25}{4} & \frac{37}{2} \\ 0 & 0 & 1 \end{pmatrix} \quad [ac] = \begin{pmatrix} 1 & \frac{9}{2} & \frac{7}{2} \\ 0 & \frac{25}{4} & \frac{59}{4} \\ 0 & 0 & 1 \end{pmatrix}$$

$$[cb] = \begin{pmatrix} 1 & 1 & 0 \\ 0 & \frac{5}{4} & \frac{7}{2} \\ 0 & 0 & 1 \end{pmatrix} \quad [ba] = \begin{pmatrix} 1 & 1 & 0 \\ 0 & \frac{5}{4} & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

This proves $R \cup S \vdash^{\mathfrak{M}(\mathbb{Q}_{\geq 0},3)} \{cb \to ba, \epsilon \to b\}$.

By another interpretation $[a] = [b] = 1, [c] = 2$ we get

$$\{cb \to ba, \epsilon \to b\} \vdash^{\mathfrak{M}(\mathbb{N},1)} \{\epsilon \to b\}.$$

□

Lemma 8. *There is no $T \subset R \cup S$ such that $(R \cup S, T) \in \mathfrak{M}(\mathbb{N})$.*

Proof. Assume there is a matrix interpretation of any size (an $\mathbb{N}$-weighted automaton with any number of states) that is weakly compatible with $R \cup S$ and strictly compatible with one of the rules from R. (It cannot be strictly compatible with S since S is non-terminating.)

We assume the automaton is reduced. All edges labelled by b are unit loops: they go from some state q to q and have weight one. The reason is that weak compatibility with S requires $[\epsilon] \geq [b]$, but $[\epsilon]$ is the unit matrix, for any interpretation $[\cdot]$.

The plan of the proof is now: we show that the interpretation of b is indeed the unit matrix (each state has a b loop), and then we derive a contradiction from that.

Consider the subset A of states that are reachable from the initial state i of the automaton by a edges. (Here and in the following, when we speak of an edge, then we mean that it has a non-zero weight.)

We claim that state q in A is also reachable from i by c edges, and has a b loop. The proof is by induction on the distance to i. Assume the transition $p \xrightarrow{a} q$ has weight > 0, and the claim holds true for p. Then we have a path $p \xrightarrow{b} p \xrightarrow{a} q$. By weak compatibility with rule $cb \to ba$, there must be a path $p \xrightarrow{cb} q$. Since b transitions are loops, this can only take the form of $p \xrightarrow{c} q \xrightarrow{b} q$.

Every state r reachable from i by any mixture of a and c steps is also in A (that is, reachable by a steps alone): assume by induction that there is a transition $q \xrightarrow{c} r$ and the claim holds true for q. Then q is in A, so there is a transition $p \xrightarrow{a} q$, thus a path $p \xrightarrow{a} q \xrightarrow{b} q \xrightarrow{c} r$. By weak compatibility with $baa \to abc$, there must be a path $p \xrightarrow{baa} r$. Since a b edge is a loop, there is some q' such that the path is $p \xrightarrow{b} p \xrightarrow{a} q' \xrightarrow{a} r$. This shows that r is in A, since it is reachable from $p \in A$ by a steps.

The final state f does also belong to A: since the interpretation was assumed to be strictly compatible with some rule $(l \to r) \in R$, there must be a path $i \xrightarrow{l} f$. Since b steps (loops) are irrelevant for reachability, the claim follows.

Since the automaton was reduced, we have shown that *each* state belongs to A, thus each state has a b loop. This implies that the interpretation of letter b is the unit matrix. Now we replace b by ϵ in R, obtaining $R' = \{aa \to ac, ca \to ac, c \to a\}$. We claim that the automaton is weakly compatible with R' and strictly compatible with at least one rule from R'. This holds true since $[baa] = [aa]$ etc., and the automaton was assumed to be weakly compatible with R and strictly compatible with at least one rule from R.

On the other hand, there is a looping R'-derivation

$$\underline{aa}a \to a\underline{ca} \to aa\underline{c} \to aaa \to \ldots$$

that uses each rule of R' infinitely often. This contradicts the fact that the automaton is strictly compatible with at least one rule of R', since this rule must be relatively terminating w.r.t. the others.

In all, this proves that the interpretation (automaton) does not exist. □

As a corollary to Lemma 7 and Lemma 8, we get

Theorem 5. $\mathfrak{M}(\mathbb{Q}_{\geq 0}, 3)^2 \setminus \mathfrak{M}(\mathbb{N})^*$ *is non-empty.*

8 Number of Proof Steps

We first recall an example that shows that two-step proofs (even of dimension two) are more powerful than one-step proofs (of any dimension). Then we generalize, and show that the "step hierarchy" is infinite. The underlying reason is derivational complexity. The following is a basic fact of linear algebra:

Lemma 9. *Let A be any finite set of square matrices of identical shape. The coefficients in a product of any k matrices from A are bounded by an exponential function of k.* □

This will be used in the following form:

Corollary 2. *For disjoint rewriting systems R and S: if there is a family of $R \cup S$-derivations*

$$d_1 : w_{1,1} \to \ldots \to w_{1,n_1}, d_2 : w_{2,1} \to \ldots \to w_{2,n_2}, \ldots$$

such that the number of R steps in d_k is not bounded by an exponential function of $|w_{k,1}|$, then $(R \cup S, S) \notin \mathfrak{M}(\mathbb{R}_{\geq 0})$.

Proof. Assume to the contrary that there is some (i, f)-pointed automaton A with the given properties: strictly compatible with R and weakly compatible with S. Then $u \to_R v$ implies $\mu(i, u, f) > \mu(i, v, f)$, and $u \to_S v$ implies $\mu(i, u, f) \geq \mu(i, v, f)$, So the number of R steps in the derivation starting in $w_{k,1}$ is bounded by $\mu(i, w_{k,1}, f)$, which is an exponential function by Lemma 9, contradicting the assumption. □

Lemma 10. *There is $R \in \mathfrak{M}(\mathbb{N}, 2)^2 \setminus \mathfrak{M}(\mathbb{R}_{\geq 0})$.*

Proof. The following example is already presented in [13]. Let $R = \{ab \to baa, cb \to bbc\}$. There are derivations (for each $k \geq 0$):

$$a^k b \to^* ba^{2k}, ab^k \to^* b^k a^{2^k}$$
$$cb^k \to^* b^{2k} c, c^k b \to^* b^{2^k} c^k$$
$$ac^k b \to^* ab^{2^k} c^k \to^* b^{2^k} a^{2^{2^k}} c^k$$

The resulting string has length 2^{2^k}, thus the derivation also took this number of steps, since each step extends the length by one.

By Lemma 9, there can be no matrix interpretation that is strictly compatible with both rules of R.

On the other hand we have $R \in \mathfrak{M}(\mathbb{N}, 2)^2$ by Example 6. □

We modify, and generalize this example. For any $n \geq 1$, define a rewriting system over alphabet $\Sigma_n = \{1, \ldots, n\}$ by

$$R_n = \{i(i-1) \to (i-1)^2 i \mid 2 \leq i \leq n\} \cup \{(i-1) \to (i-2) \mid 3 \leq i \leq n\}.$$

For $n \geq 2$, this system has $2n - 3$ rules. Note that R_1 is empty.

Example 10. $R_3 = \{32 \to 223, 21 \to 112, 2 \to 1\}$. □

Lemma 11. *For each i and k, there is a R_n-derivation from $i^k(i-1)$ to some word containing $(i-1)^{2^{k-1}}(i-2)$ as a factor and using each of the rules $i(i-1) \to (i-1)^2 i$ and $(i-1) \to (i-2)$ at least 2^{k-1} times.*

Proof. For each l, we have $i(i-1)^l \to^l (i-1)^{2l} i$, and by iteration,

$$i^k(i-1) \to^{2^{k+1}-1} (i-1)^{2^k} i^k.$$

Now we apply rule $(i-1) \to (i-2)$ for 2^{k-1} times to get

$$(i-1)^{2^{k-1}}(i-2)^{2^{k-1}} i^k.$$

□

Using Lemma 11 repeatedly, we get

Lemma 12. *For each i and k, there is a R_n-derivation from $i^k(i-1)$ using at least $\exp(\exp(k))$ steps of each rule $j(j-1) \to (j-1)^2 j$ and $(j-1) \to (j-2)$, for $j < i$.* □

Lemma 13. *If a matrix interpretation is weakly compatible with R_n and strictly compatible with some subset $S \subseteq R_n$, then $R_{n-1} \cap S = \emptyset$.*

Proof. By Lemma 12, there is a family of derivations that uses all rules in R_{n-1} more than exponentially often. By Corollary 2, the claim follows. □

Lemma 14. $R_{n+2} \notin \mathfrak{M}(\mathbb{R}_{\geq 0})^n$.

Proof. $R_2 \notin \mathfrak{M}(\mathbb{R}_{\geq 0})^0$ since R_2 is non-empty. By Lemma 13, if $R_{n+1} \overset{\mathfrak{M}(\mathbb{R}_{\geq 0})}{\not\to} R'$, then $R_n \subseteq R'$. Then the claim follows by induction. □

Lemma 15. $R_{n+1} \in \mathfrak{M}(\mathbb{N}, 2)^n$.

Proof. $R_1 \in \mathfrak{M}(\mathbb{R}_{\geq 0}, 2)^0$ since R_1 is empty. The following interpretation

$$[n] = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, [n-1] = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \text{for } j \leq n-2 : [j] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

shows $R_n \vdash^{\mathfrak{M}(\mathbb{N},2)} R_{n-1}$. Then the claim follows by induction. □

We obtain as a corollary:

Theorem 6. *Each inclusion* $\mathfrak{M}(\mathbb{R}_{\geq 0})^s \subset \mathfrak{M}(\mathbb{R}_{\geq 0})^{s+1}$ *is strict. The "proof length hierarchy"* $(\mathfrak{M}(\mathbb{R}_{\geq 0})^s)_{s=1,2,\ldots}$ *is infinite.*

9 Discussion

Summary. Termination proofs by weighted (word and tree) automata are being investigated only since 2006. (resp. 2003, if we include the Match Bounds method, which later turned out to be related to the Min/Max semiring.) The focus of investigation mainly was the construction of automata, with the goal of actually implementing and running the algorithms. This has been achieved rather successfully: the various "matrix methods" play a decisive role in the regular Termination Competitions.

With the present paper, we start a systematic investigation into the expressiveness of these methods as *proof systems.*

To this end, we have defined a two-dimensional hierarchy $\mathfrak{M}(W, d)^s$ for termination proofs for string rewriting via weighted word automata, and we proved that the hierarchy is infinite in both directions.

Still, we have no exact information on which levels are actually inhabited (notice the "gap" from d to d' in Lemma 4), and which levels (if any) are decidable. These questions remain as challenging open problems. Other extensions are at hand, and we list a few.

Decidability. As noted in the proof of Lemma 5, existence of a one-dimensional interpretation is equivalent to the feasibility of a system of linear inequalities. Therefore, $\mathfrak{M}(\mathbb{R}_{\geq 0}, 1)$ is decidable.

For larger dimensions, the weak and strong compatibility conditions give rise to a system of inequalities between polynomials, where the unknowns are the matrix entries (the weights of the automaton transitions). Then we can use Tarski's decision method [16], and obtain that for each d, $\mathfrak{M}(\mathbb{R}_{\geq 0}, d)$ is decidable.

In fact if the system of polynomial (in)equalities has a solution, then it also has a solution in algebraic numbers. So we don't really need real numbers: for each d, $\mathfrak{M}(\mathrm{Alg}_{\geq 0}, d) = \mathfrak{M}(\mathbb{R}_{\geq 0}, d)$.

Except for these immediate observations, we have no information (and no intuition) on decidability of any $\mathfrak{M}(W, d)$.

Non-strict semirings. One can use semirings with non-strict addition for termination, e.g. the max/plus semiring, or the max/min semiring [17]. Again, a corresponding hierarchy can be defined but it needs different methods than presented here. If we try the construction of Section 5, using a suitable polynomial identity $\sum[l_i] = \sum[r_i]$ in the arctic semiring, we can no longer infer from $\forall i : [l_i] \geq [r_i]$ that $\forall i : [l_i] = [r_i]$, since arctic addition is not strictly monotonic in its arguments. For the proof step hierarchy we cannot use the methods of Section 8, because of the following: Arctic interpretations give a linear bound on derivational complexity, and by the reasoning in Lemma 6, even a combination of such interpretations might not achieve more than linear derivation lenghts. So, the "arctic termination hierarchy" is a subject of further study.

Term rewriting. The method of interpretation via weighted automata has been generalized to term rewriting [6]. The definition of our hierarchy can be generalized as well. Still we note that matrix interpretations for term rewriting use a rather restricted form of weighted tree automata.

Parallel composition of proofs. Our hierarchy uses the concept of combining termination proofs *sequentially*. There are methods of proving termination that correspond to a *parallel* composition: after the Dependency Pairs transformation [1], the resulting relative termination problem can be decomposed into several independent sub-problems, corresponding to the strictly connected components of the dependency graph [12]. In all, a termination proof thus gets a tree structure. While we presently compare proof sequences by length, proof trees should be compared structurally, e.g. with respect to embedding.

References

[1] Arts, Thomas and Giesl, Jürgen. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.

[2] Berstel, Jean and Reutenauer, Christophe. *Noncommutative Rational Series With Applications.* `http://www-igm.univ-mlv.fr/~berstel/LivreSeries/LivreSeries.html`, 2009.

[3] Book, Ronald V and Otto, Friedrich. *String Rewriting Systems.* Springer, 1993.

[4] Drensky, Vesselin and Formanek, Edward. *Polynomial Identity Rings.* Birkhäuser, 2004.

[5] Droste, Manfred, Kuich, Werner, and Vogler, Heiko, editors. *Handbook of Weighted Automata.* Springer, 2009. in preparation.

[6] Endrullis, Jörg, Waldmann, Johannes, and Zantema, Hans. Matrix interpretations for proving termination of term rewriting. In Furbach, Ulrich and

Shankar, Natarajan, editors, *IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 574–588. Springer, 2006.

[7] Fuchs, Laszlo. *Partially Ordered Algebraic Systems.* Addison-Wesley, 1962.

[8] Gebhardt, Andreas, Hofbauer, Dieter, and Waldmann, Johannes. Matrix evolutions. In Hofbauer, Dieter and Serebrenik, Alexander, editors, *Proc. Workshop on Termination, Paris*, 2007.

[9] Gebhardt, Andreas and Waldmann, Johannes. Weighted automata define a hierarchy of terminating string rewriting systems. In Droste, Manfred and Vogler, Heiko, editors, *Proc. Weighted Automata Theory and Applications, Dresden*, pages 34–35, 2008.

[10] Geser, Alfons. *Relative termination.* Dissertation, Fakultät für Mathematik und Informatik, Universität Passau, Germany, 1990. 105 pages. Also available as: Report 91-03, Ulmer Informatik-Berichte, Universität Ulm, 1991.

[11] Golan, Jonathan S. *Semirings and their Applications.* Kluwer, 1999.

[12] Hirokawa, Nao and Middeldorp, Aart. Dependency pairs revisited. In van Oostrom, Vincent, editor, *RTA*, volume 3091 of *Lecture Notes in Computer Science*, pages 249–268. Springer, 2004.

[13] Hofbauer, Dieter and Waldmann, Johannes. Termination of string rewriting with matrix interpretations. In Pfenning, Frank, editor, *RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 328–342. Springer, 2006.

[14] Kanel-Belov, Alexei and Rowen, Louis Halle. *Computational Aspects of Polynomial Identities.* AK Peters, 2005.

[15] Kuich, Werner. Semirings and formal power series. In *Handbook of Formal Languages*, volume 1, pages 609–677. Springer, 1997.

[16] Tarski, Alfred. A decision method for elementary algebra and geometry. Manuscript. Santa Monica, CA: RAND Corp., 1948.

[17] Waldmann, Johannes. Weighted automata for proving termination of string rewriting. *Journal of Automata, Languages and Combinatorics*, 12(4):545–570, 2007.

[18] Zantema, Hans. Termination. In Terese, editor, *Term Rewriting Systems*, pages 181–259. Cambridge University Press, 2003.

Received 1st August 2008

Acta Cybernetica 19 (2009) 313–356.

Statistical Language Models within the Algebra of Weighted Rational Languages

Thomas Hanneforth* and Kay-Michael Würzner*

Abstract

Statistical language models are an important tool in natural language processing. They represent prior knowledge about a certain language which is usually gained from a set of samples called a *corpus*. In this paper, we present a novel way of creating N-gram language models using weighted finite automata. The construction of these models is formalised within the algebra underlying weighted finite automata and expressed in terms of weighted rational languages and transductions. Besides the algebra we make use of five special constant weighted transductions which rely only on the alphabet and the model parameter N. In addition, we discuss efficient implementations of these transductions in terms of *virtual constructions*.

Keywords: computational linguistics, weighted rational transductions, statistical language modeling, N-gram models, weighted finite-state automata

1 Introduction

Weighted finite-state acceptors (WFSA) provide a convenient way to compactly represent N-gram language models (cf. [3]) since they admit equivalence transformations like determinisation and minimisation [22] which compress common prefixes and suffixes without changing the counts or probabilities associated with an individual N-gram. Moreover, it is possible to represent all sub-distributions of M-grams (with $1 \leq M < N$) simultaneously with almost no additional space.

The usual way is to construct the language models on the basis of the manipulation of states and transitions. Since the models are also required to be robust, it is necessary to reserve some probability mass for unseen N-grams. This is commonly achieved by combining a discounting method with a back-off [17] or interpolation mechanism [15]. The adjusted probabilities are then reassigned for each N-gram to existing or newly created transitions. The finite automata thus merely serve as a data structure.

*University of Potsdam, E-mail: {tom,wuerzner}@ling.uni-potsdam.de

In this paper, we present an approach which treats the creation of N-gram models as a problem of modifying weighted languages rather than states and transitions. In particular, we only use operations from the algebra of weighted regular languages (WRLs) and transductions (WRTs) like union and intersection to get from a set of samples to a robust back-off model. Such an algebraic formalisation has – at least to our knowledge – never been done before.

The results outlined in the remainder are by now mainly of theoretical interest. We do not aim to replace the many excellent statistical toolkits by the machinery proposed here. This work is rather a "case study" in viewing an important tool in natural languages processing from a theoretical viewpoint. As such, we describe it in a self-contained form.

This article is organised as follows: In Section 2, we will recall the notion of language models in general and N-gram models in particular (may be skipped by readers familiar with the topic). Section 3 introduces the formal preliminaries and establishes the notation. The subsequent sections 4-7 deal with the creation of N-gram and back-off models from scratch in the manner explained above. Matters of complexity and implementation are discussed in each section. Proofs of correctness of the outlined methods have been put in the appendix for reasons of readability.

2 Language Models

Language modeling is the task of assigning a probability to sequences of words. $\Pr(w)$ is the prior probability of the sequence of words w. Language models are used in many applications in natural language processing such as speech recognition, machine translation, optical character recognition or part-of-speech tagging. See [16] for an introduction to these topics and their relation to language models.

Using *conditional probabilities*, the joint probability of a sequence of words can be decomposed as: [1]

$$\Pr(w_1^m) = \Pr(w_1) \prod_{i=2}^{m} \Pr(w_i | w_1^{i-1}) \ . \tag{1}$$

The interdependencies of words are reflected by assuming that the occurrence of a word is a consequence of the occurrence of its predecessors. The conditional probability of a sequence of words can be computed by normalising its frequency relative to the frequency of its history ($\mathrm{C}(s)$ denotes the number of occurrences of a substring s in w, Σ refers to a finite alphabet and the sum operator, respectively):

$$\Pr(w_i | w_1^{i-1}) = \frac{\mathrm{C}(w_1^{i-1} \cdot w_i)}{\sum_{a \in \Sigma} \mathrm{C}(w_1^{i-1} \cdot a)} \ . \tag{2}$$

[1] We denote a substring $w_i \ldots w_j$ with $j \geq i$ in a more compact way by w_i^j. If $i = j$, we omit the superscript and write simply w_i for the i^{th} character of w (starting at 1). If the subscript exceeds the superscript, we implicitly denote the empty string ε.

Frequency information is obtained from a *(text) corpus* which is usually defined as a large collection of (annotated or unannotated) texts. In the remainder of this article, we use the term *corpus* as denoting a finite disjunction of *sentences*. A sentence roughly corresponds to its linguistic definition, but is not limited to that. Some big natural language corpora are the DWDS-Korpus [11] or the Brown corpus [18]. A *probability distribution* with respect to Σ^* is an assignment of probabilities to the strings in Σ^* such that all the individual string probabilities sum up to one.

2.1 N-Gram Models

N-gram models are the most widely used type of language models. Their success is based in their simplicity: They can be derived unsupervised, by just counting sequences of words in a corpus and computing their relative frequency.

In the field of language modeling, an N-gram is a sequence of N elements taken from a fixed and finite alphabet Σ, for example letters [29], words [3], morphemes, etc.

In order to limit the number of possible contexts of a word, it is assumed that sequences of words form *Markov chains* [20]. Thus, only the last $N-1$ words (sometimes also called the *history* of w_i) affect the word w_i:

$$\Pr(w_i|w_1^{i-1}) \approx \Pr(w_i|w_{i-(N-1)}^{i-1}) \ . \tag{3}$$

The number of possible contexts is then the size of the alphabet to the power of $N-1$ and therefore finite. The boundary case at the beginning of the sentence is handled by $N-1$ beginning-of-sentence markers (see Section 6 for details).

2.2 Smoothing

While theoretically possible, one will never find all potential N-grams in a corpus in practice. The common solution to this problem is *smoothing*: Probability mass is assigned to unseen events and/or other distributions which account for those events are consulted. For N-gram models, this means to change the model in such a way that it assigns a probability to any combination of N words of the vocabulary, deals adequately with out-of-vocabulary items and, is still a *probabilistic* model.

Probabilistic N-gram models are characterised by the property that for every context the probabilities of possible continuations sum up to one ($h \in \Sigma^{N-1}$):

$$\forall h \sum_{w_i} \Pr(w_i|h) = 1 \ . \tag{4}$$

Many different smoothing methods for different purposes are available (cf. [6] for a detailed summary and comparison of important smoothing methods).

For the purpose of this work, we recall the notions of *discounting* and *back-off* smoothing.

2.2.1 Discounting

The main idea behind this class of procedures is to redistribute probability mass from seen to unseen events. A simple but effective discounting algorithm is the so called *Witten-Bell discounting*, referring to method C in [30]. Witten-Bell discounting is based on the intuition that the probability of novel events decreases with the number of different events that are observed in the corpus. To implement this idea, the frequencies of the N-grams are normalised by the number of different N-grams sharing the same $(N-1)$-gram prefix. The number of different events in an event space is often called the number of *types*.

Definition 1 (Witten-Bell Type Number). *Let* T *be a function* $\Sigma^* \to \mathbb{N}$:

$$\mathrm{T}(w_{i-N+1}^{i}) = \sum_{a \in \Sigma, \mathrm{C}(w_{i-N+1}^{i-1} \cdot a) \neq 0} 1 \ .$$

Definition 2 (Witten-Bell Token Number). *Let* N *be a function* $\Sigma^* \to \mathbb{N}$:

$$\mathrm{N}(w_{i-N+1}^{i}) = \sum_{a \in \Sigma} \mathrm{C}(w_{i-N+1}^{i-1} \cdot a) \ .$$

With the help of the functions T and N it is possible to discount frequencies, denoted by $\tilde{\mathrm{C}}$:

$$\tilde{\mathrm{C}}(w_{i-N+1}^{i}) = \mathrm{C}(w_{i-N+1}^{i}) \frac{\mathrm{N}(w_{i-N+1}^{i})}{\mathrm{N}(w_{i-N+1}^{i}) + \mathrm{T}(w_{i-N+1}^{i})} \ . \tag{5}$$

Adjusted probabilities $\tilde{\Pr}$ can be computed from $\tilde{\mathrm{C}}$ [16]. The freed frequency mass is computed by:

$$\begin{aligned} &\sum_{w_{i-N+1}^{i} \in \Sigma^N} \mathrm{C}(w_{i-N+1}^{i}) - \tilde{\mathrm{C}}(w_{i-N+1}^{i}) \\ &= \sum_{w_{i-N+1}^{i} \in \Sigma^N} \mathrm{C}(w_{i-N+1}^{i}) \ \frac{\mathrm{T}(w_{i-N+1}^{i})}{\mathrm{N}(w_{i-N+1}^{i}) + \mathrm{T}(w_{i-N+1}^{i})} \ . \end{aligned} \tag{6}$$

2.2.2 Smoothing by Combining Different Distributions

Spreading saved probability mass equally among all unseen events is often too simple. It seems reasonable to take different distributions into account. A common way of doing that is the *back-off* strategy [17] which recursively uses the $(N-1)$-gram distribution whenever the N-gram distribution assigns a zero probability. Equation (7) formalises this behavior by defining the *back-off probability* $\hat{\Pr}$:

$$\begin{aligned} \hat{\Pr}(w_i | w_{i-N+1}^{i-1}) = &\tilde{\Pr}(w_i | w_{i-N+1}^{i-1}) \\ &+ \phi(\tilde{\Pr}(w_i | w_{i-N+1}^{i-1})) \\ &\cdot \alpha(w_{i-N+1}^{i-1}) \hat{\Pr}(w_i | w_{i-N+2}^{i-1}) \ . \end{aligned} \tag{7}$$

Function ϕ indicates the need for backing-off to the immediately lower ordered distribution:

$$\phi(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ 1 & \text{otherwise .} \end{cases} \tag{8}$$

This ensures that one of the summands in Equation (7) will always be equal to 0. $\tilde{\Pr}$ and α depend on the particular discounting algorithm and represent the adjusted probabilities and the normalised freed probability mass, respectively:

$$\alpha(h) = \begin{cases} 1 - \sum_{w_i} \tilde{\Pr}(w_i|h) & \text{if } \mathrm{C}(h) > 0 \\ 1 & \text{otherwise .} \end{cases} \tag{9}$$

The second case in Equation (9) covers events where the $(N-1)$-gram history is not available. The lower ordered distribution is used unweighted in such cases. Since lower ordered distributions are probabilistic by definition, the whole model keeps this property.

The back-off recursion is terminated either by the (undiscounted) unigram distribution

$$\hat{\Pr}(w_i) = \Pr(w_i) \ . \tag{10}$$

or by a uniform distribution which handles out-of-vocabulary items. Such a uniform distribution involves a non-probabilistic model, since any number of out-of-vocabulary items is possible:

$$\hat{\Pr}(\varepsilon) = \Pr_{unif}(\varepsilon) = \frac{1}{\sum_{b \in \Sigma} 1} \ . \tag{11}$$

Back-off smoothing is compatible with all discounting algorithms. We use Witten-Bell discounting as explained above.

3 Formal Preliminaries

In this section, we define the formal apparatus used in the remainder of this article. We start with the notion of a *semiring*, define weighted rational languages and transductions, move to the definition of weighted finite-state acceptors and transducers and a number of operations defined on them and finally clarify the relationship between weighted languages on the one and finite automata on the other hand.

3.1 Semirings

The weights of languages, transductions and automata are expressed in terms of a semiring. The advantage in doing so lies in the abstraction and well-definedness of operations and algorithms for different types of weights (e.g. [19, 25, 24]).

Definition 3 (Semiring). *A structure* $\mathcal{K} = \langle \mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1} \rangle$ *is a semiring if*

1. $\langle \mathbb{K}, \oplus, \overline{0} \rangle$ *is a commutative monoid with* $\overline{0}$ *as the identity element for* $\oplus$,
2. $\langle \mathbb{K}, \otimes, \overline{1} \rangle$ *is a monoid with* $\overline{1}$ *as the identity element for* $\otimes$,
3. $\otimes$ *distributes over* $\oplus$ *(distribution of one operation over another will be denoted by* $\succ$*, e.g.* $\otimes \succ \oplus$*) , and*
4. $\overline{0}$ *is an annihilator for* $\otimes$*:* $\forall a \in \mathbb{K}, a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$.

Examples for semirings are the *boolean* semiring $\mathcal{B} = \langle \{0,1\}, \vee, \wedge, 0, 1 \rangle$, the *real* semiring $\mathcal{R} = \langle \mathbb{R} \cup \{\infty\}, +, \cdot, 0, 1 \rangle$, the *log* semiring $\mathcal{L} = \langle \mathbb{R} \cup \{\infty\}, +_{\log}, +, \infty, 0 \rangle$[2] or the *tropical* semiring $\mathcal{T} = \langle \mathbb{R}^+ \cup \{\infty\}, \min, +, \infty, 0 \rangle$. A special significance in the remainder of this work lays on the *probability* semiring $\mathcal{P} = \langle \mathbb{R}^+ \cup \{\infty\}, +, \cdot, 0, 1 \rangle$ since its properties make it suitable for representing probabilities.[3]

To be well-defined, some operations on languages and automata demand particular properties of the used semirings. See [19] for a detailed summary on semirings and their properties. For the scope of this article, we need the definitions of *idempotency*, *divisibility*, *commutativity* and *completeness*.

Definition 4 (Idempotent Semiring). *A semiring* $\mathcal{K}$ *is called idempotent if* $a \oplus a = a$ *for all* $a \in \mathbb{K}$.

Definition 4 means that in case of non-idempotent semirings the $\oplus$ operation is effectively additive in a sense that it sums weights. The probability and the log semiring are non-idempotent.

Definition 5 (Division Semiring). *A semiring* $\mathcal{K}$ *is a division semiring iff* $\forall a \in \mathbb{K} \setminus \{\overline{0}\}$, $\exists! b \in \mathbb{K}$ *such that* $a \otimes b = \overline{1}$.

Divisibility (cf. [9]) is a formalisation of the demand for closure under *multiplicative inversion* needed for division of elements in $\mathbb{K}$. This property is adapted from a special class of *rings* called the *divisible rings*.

Definition 6 (Commutative Semiring). *A semiring is said to be commutative when the* $\otimes$ *operation is commutative; that is* $\forall a, b \in \mathbb{K}, a \otimes b = b \otimes a$.

The requirement that sums of an infinite number of elements are well defined is expressed as *completeness* (e.g. [10]).

Definition 7 (Complete Semiring). *A semiring* $\mathcal{K}$ *is called complete if it is possible to define sums for all families* $(a_i | i \in I)$ *of elements in* $\mathbb{K}$*, where* I *is an arbitrary index set, such that the following conditions are satisfied:*

[2] $a +_{\log} b =_{def} -\log(2^{-a} + 2^{-b})$

[3] The terms 'probability semiring' and 'real semiring' are interchanged freely in the corresponding literature. The following distinction seems sensible: Since real numbers can be both positive and negative, the real semiring should be defined over $\mathbb{R}$. Probability on the other hand will always be positive, thus in $\mathbb{R}^+$.

(i) $\bigoplus_{i\in\emptyset} a_i = \overline{0}$, $\bigoplus_{i\in\{j\}} a_i = a_j$, $\bigoplus_{i\in\{j,k\}} a_i = a_j \oplus a_k$ *for* $j \neq k$,

(ii) $\bigoplus_{j\in J}(\bigoplus_{i\in I_j} a_i) = \bigoplus_{i\in I} a_i$, *if* $\bigcup_{j\in J} I_j = I$ *and* $I_j \cap I_{j'} = \emptyset$ *for* $j \neq j'$,

(iii) $\bigoplus_{i\in I}(c \otimes a_i) = c \otimes (\bigoplus_{i\in I} a_i)$, $\bigoplus_{i\in I}(a_i \otimes c) = (\bigoplus_{i\in I} a_i) \otimes c$.

In the following, we restrict our attention to commutative, divisible, complete and non-idempotent semirings. For better transparency, we primarily use the real semiring $\mathcal{R}$ and the probability semiring $\mathcal{P}$ in definitions, lemmas and proofs, but also use the more general semiring notation with $\oplus$ and $\otimes$.[4]

We will denote semirings with capital letters in calligraphic character style like $\mathcal{P}$, $\mathcal{K}$.

3.2 Weighted Rational Languages and Transductions

Every formal language can be represented as a weighted language.

Definition 8 (Weighted Language). *A weighted language* $\mathcal{L}$ *is a mapping* $\Sigma^* \to \mathcal{K}$, *where* Σ *denotes a finite set of symbols (called the alphabet) and* $\mathcal{K}$ *a semiring.*

This definition applies to all formal languages. The different types of languages are distinguished by the operations that are allowed to construct the subset of Σ^* from the singletons in Σ (see below).

Definition 9 (Weighted Transduction). *A weighted transduction* $\mathcal{S}$ *is a mapping* $\Sigma^* \times \Gamma^* \to \mathcal{K}$, *where* Σ *and* Γ *denote finite sets of symbols (called the input and the output alphabet, resp.) and* $\mathcal{K}$ *a semiring.*

Weighted rational languages (WRL) and *weighted rational transductions* (WRT) are a proper subset of the weighted languages and transductions. They can be constructed from singletons in a finite alphabet Σ using *scaling*, *union*, *concatenation*, *composition* and *closure* [26]. In addition to these, we use a set of operations on WRLs and WRTs summarised in Table 1.

Definition 10 equates any WRL with its identity transduction.

Definition 10 (Identity Transduction). *Given a WRL* $\mathcal{L} : \Sigma^* \to \mathcal{K}$, *its identity transduction* $ID(\mathcal{L}) : \Sigma^* \times \Sigma^* \to \mathcal{K}$ *is defined as:*

$$\forall x, y \in \Sigma^*,\ ID(\mathcal{L})(x, y) = \begin{cases} \mathcal{L}(x) & \text{if } x = y \\ \overline{0} & \text{otherwise} \end{cases} .$$

An often used complex operation is *application*:

Definition 11 (Application). *The application of a WRT* $\mathcal{S} : \Sigma^* \times \Gamma^* \to \mathcal{K}$ *to a WRL* $\mathcal{L} : \Sigma^* \to \mathcal{K}$ *is a mapping* $\mathcal{S}[\mathcal{L}] : \Gamma^* \to \mathcal{K}$ *defined by*

$$\forall y \in \Gamma^*,\ \mathcal{S}[\mathcal{L}](y) = \bigoplus_{x\in\Sigma^*} \mathcal{L}(x) \otimes \mathcal{S}(x, y) .$$

[4] In practice, $\mathcal{P}$'s isomorphic counter part, the log semiring $\mathcal{L}$ would be used instead for reasons of numerical stability.

Table 1: Operations on WRLs and WRTs
Let $\mathcal{S}$: $\Sigma^* \times \Delta^* \to \mathcal{K}$, and $\mathcal{Q}$: $\Delta^* \times \Gamma^* \to \mathcal{K}$, denote two WRTs and let $\mathcal{L}_1$: $\Sigma^* \to \mathcal{K}$, and $\mathcal{L}_2$: $\Sigma^* \to \mathcal{K}$, denote two WRLs.[a] Let a, b and c, d be chosen from the same alphabet (augmented with ε), respectively. For $\mathcal{S}$ (also $\mathcal{S}_1, \mathcal{S}_2$), let the operands x and y range over Σ^* and Δ^*, resp. For $\mathcal{Q}$, let x and y range over Δ^* and Γ^*, resp. For $\mathcal{L}_1$ and $\mathcal{L}_2$, $x, y \in \Sigma^*$.

singleton	$\{(a,c)\}(b,d)$	$=$	$\overline{1}$ if $a = b$ and $c = d$, $\overline{0}$ otherwise
singleton	$\{a\}(b)$	$=$	$\overline{1}$ if $a = b$, $\overline{0}$ otherwise
union (sum)	$(\mathcal{S}_1 \cup \mathcal{S}_2)(x,y)$	$=$	$\mathcal{S}_1(x,y) \oplus \mathcal{S}_2(x,y)$
concatenation	$(\mathcal{S}_1 \cdot \mathcal{S}_2)(x,y)$	$=$	$\bigoplus_{tu=x, vw=y} \mathcal{S}_1(t,v) \otimes \mathcal{S}_2(u,w)$
scaling	$k\mathcal{Q}(x,y)$	$=$	$k \otimes \mathcal{Q}(x,y) \quad (k \in \mathcal{K})$
power	$\mathcal{Q}^0(\varepsilon,\varepsilon)$	$=$	$\overline{1}$
	$\mathcal{Q}^0(x \neq \varepsilon, y \neq \varepsilon)$	$=$	$\overline{0}$
	$\mathcal{Q}^{n+1}(x,y)$	$=$	$(\mathcal{Q} \cdot \mathcal{Q}^n)(x,y)$
closure	$\mathcal{Q}^*(x,y)$	$=$	$\bigoplus_{k \geq 0} \mathcal{Q}^k(x,y)$
composition	$(\mathcal{S} \circ \mathcal{Q})(x,y)$	$=$	$\bigoplus_{z \in \Delta^*} \mathcal{S}(x,z) \otimes \mathcal{Q}(z,y)$
1^{st} *projection*	$\pi^1(\mathcal{S})(x)$	$=$	$\bigoplus_{y \in \Delta^*} \mathcal{S}(x,y)$
2^{nd} *projection*	$\pi^2(\mathcal{S})(y)$	$=$	$\bigoplus_{x \in \Sigma^*} \mathcal{S}(x,y)$
crossproduct	$(\mathcal{L}_1 \times \mathcal{L}_2)(x,y)$	$=$	$\mathcal{L}_1(x) \otimes \mathcal{L}_2(y)$
intersection	$(\mathcal{L}_1 \cap \mathcal{L}_2)(x)$	$=$	$\mathcal{L}_1(x) \otimes \mathcal{L}_2(x)$

[a]Using the identity transduction from Definition 10, the operations union, concatenation, power, scaling, and closure also apply to weighted rational languages.

Application is a short-cut for composing the identity transduction of $\mathcal{L}$ with $\mathcal{S}$ and taking the 2^{nd} projection afterwards.

Definition 12 (Language Projection). *Given a WRL $\mathcal{L} : \Sigma^* \to \mathcal{K}$, the language projection of $\mathcal{L}$ – denoted by $\pi^L(\mathcal{L})$ – is defined as*

$$\forall x \in \Sigma^*,\ \pi^L(\mathcal{L})(x) = \begin{cases} \overline{1} & \text{if } \mathcal{L}(x) \neq \overline{0} \\ \overline{0} & \text{otherwise} \end{cases}.$$

Since language projection is an operation which only replaces the weights of its operand, WRLs and WRTs are closed under it (see the result for *length preserving* homomorphisms in [8]).

Definition 13 ($\otimes$-negation). *Given a WRL $\mathcal{L} : \Sigma^* \to \mathcal{K}$ over a division semiring $\mathcal{K}$, the $\otimes$-negation of $\mathcal{L}$ – denoted by $\mathcal{L}^{-\overline{1}}$ – is defined as*

$$\forall x \in \Sigma^*,\ \mathcal{L}^{-\overline{1}}(x) = \begin{cases} a & \text{if } \mathcal{L}(x) \neq \overline{0} \ \wedge\ a \otimes \mathcal{L}(x) = \overline{1} \\ \overline{0} & \text{otherwise .} \end{cases}$$

Further on, we use capital script letters like $\mathcal{L}$, $\mathcal{P}$ to denote weighted languages and transductions.

3.3 Weighted Finite-State Automata

Every WRL and every WRT can be represented by at least one weighted finite-state acceptor or transducer, respectively.

Definition 14 (WFSA). *A* weighted finite-state acceptor *(henceforth WFSA, cf. [24]) $\mathfrak{A} = \langle \Sigma, Q, q_0, F, E, \lambda, \rho \rangle$ over a semiring $\mathcal{K}$ is a 7-tuple with*

1. *Σ, the finite input alphabet,*
2. *Q, the finite set of states,*
3. *$q_0 \in Q$, the start state,*
4. *$F \subseteq Q$, the set of final states,*
5. *$E \subseteq Q \times Q \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{K}$, the set of transitions,*
6. *$\lambda \in \mathcal{K}$, the initial weight, and*
7. *$\rho : F \to \mathcal{K}$, the final weight function mapping final states to elements in $\mathcal{K}$.*

An extension of WFSAs are the *weighted finite-state transducers.*

Definition 15 (WFST). *A* weighted finite-state transducer *(henceforth WFST) $\langle \Sigma, \Delta, Q, q_0, F, E, \lambda, \rho \rangle$ over a semiring $\mathcal{K}$ is a 8-tuple with*

1. *Σ, Q, q_0, F, λ and ρ are defined in the same manner as in the case of WFSAs,*
2. *Δ, the finite output alphabet, and*
3. *$E \subseteq Q \times Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times \mathcal{K}$, the set of transitions.*

The weight assigned by a WFSA $\mathfrak{A}$ to a string $x \in \Sigma^*$ is determined by Definition 16.

Definition 16 (Weight of a String). *Let $\mathfrak{A} = \langle \Sigma, Q, q_0, F, E, \lambda, \rho \rangle$ be a WFSA over a semiring $\mathcal{K}$. Let π be a path in $\mathfrak{A}$, that is, a sequence of adjacent transitions. Let $n[\pi]$ denote the state reached at the end of π. Let $\Pi(Q_1, x, Q_2)$ denote the set of all paths from $q_1 \in Q_1$ to $q_2 \in Q_2$ labeled with $x \in \Sigma^*$. Let $\omega(\pi)$ denote the $\otimes$-multiplication of the weights of the transitions along the path π. The weight assigned to a string $x \in \Sigma^*$ by $\mathfrak{A}$, denoted by $[\![x]\!]^{\mathfrak{A}}$, is defined as:*

$$[\![x]\!]^{\mathfrak{A}} = \bigoplus_{\pi \in \Pi(\{q_0\}, x, F)} \lambda \otimes \omega(\pi) \otimes \rho(n(\pi)) \ .$$

A WFSA is called *unambiguous*, if there is for each input string x at most a single path in $\mathfrak{A}$. As a special case, each state q in a *deterministic* WFSA has at most a single target state for each $a \in \Sigma$. Note that in case of unambiguous/deterministic WFSAs, the $\oplus$-operation in Definition 16 has no effect, since there is for every input string only a single path from q_0 to a final state.

In addition to the automata-algebraic operations like union, intersection, concatenation etc., we use three equivalence operations, e.g. operations which only change the structure of a WFSA but not the weighted language it accepts, parametrised with respect to a semiring $\mathcal{K}$: $rm\text{-}\varepsilon_{\mathcal{K}}$ for ε-removal, $det_{\mathcal{K}}$ for determinisation of WFSAs, and $min_{\mathcal{K}}$ for minimisation. We omit the subscript for the semiring if it is understood from the context.

If $\mathcal{K}$ is a divisible semiring, we denote by $neg^{\otimes}_{\mathcal{K}}$ the operation, which replaces the initial weight λ and each transition and final state weight a of a WFSA $\mathfrak{A}$ by its multiplicative inverse, denoted by $\lambda^{-\overline{1}}$ and $a^{-\overline{1}}$ respectively. Note that $\mathfrak{A}$ must be at least unambiguous to obtain the correct result corresponding to Definition 13. Although not every WFSA can be determinised [21], those WFSAs to which we apply $neg^{\otimes}_{\mathcal{K}}$ have an equivalent deterministic counterpart.

Typographically, we will render acceptors and transducers with letters in Gothic type, for example $\mathfrak{E}$, $\mathfrak{K}$.

4 N-Gram Counting

As shown in Section 2, frequencies of events are necessary for creating N-gram word models. This section shows how to obtain these frequencies.

4.1 Text Corpora as Weighted Finite-State Automata

Text corpora can be easily represented as acyclic weighted finite state acceptors over the real semiring. This approach is advantageous since acyclic WFSAs always admit equivalence transformations like determinisation and minimisation [21]. Fig. 1 shows a WFSA $\mathfrak{K}$ constructed from a toy corpus.[5]

[5] We adopt the convention that transition labels are of the form a/w in case of acceptors and $a : b/w$ when depicting transducers: $a \in \Sigma \cup \{\varepsilon\}$ denotes the input symbol of the transition, $b \in \Delta \cup \{\varepsilon\}$ is its output symbol and $w \in K$ its weight. In the context of an WFST, a transition labeled with a stands for the identity transduction $a : a$. Similar, the final weight $\rho(p)$ assigned to

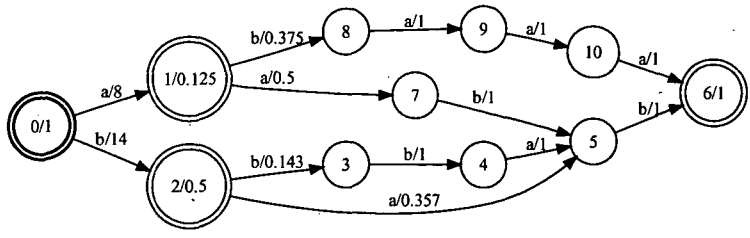


Figure 1: A toy corpus over $\Sigma = \{a, b\}$ represented as a WFSA $\mathfrak{K}$.

The number of occurrences of a given sentence s can be computed along Definition 16; for example $[\![aabb]\!]^{\mathfrak{K}} = 1 \cdot 8 \cdot 0.5 \cdot 1 \cdot 1 \cdot 1 = 4$.

4.2 N-gram Counting

An approach for counting N-grams with WFSTs has been proposed in [2]. We adopt this approach and repeat the resulting definitions using the notation introduced in Section 3. For the purpose of counting N-grams, a special transducer which realises a rational transduction $\mathcal{F} : \Sigma^* \times \Sigma^* \to \mathcal{R}$ is used:

$$\forall x, y \in \Sigma^*,\ \mathcal{F}(x, y) = ((\Sigma \times \{\varepsilon\})^* \cdot\ ID(\mathcal{L})\ \cdot (\Sigma \times \{\varepsilon\})^*)\ (x, y) \qquad (12)$$

where $\mathcal{L}$ is a WRL mapping Σ^* to $\mathcal{R}$, such that the number of strings x with $\mathcal{L}(x) \neq 0$ is finite. In the case of N-gram counting, the domain of $\mathcal{L}$ needs to be Σ^N (in which case we write $\mathcal{F}_N(x, y)$). To gain some information about which words occurred at the beginning or end of a sentence in the corpus, we augment the alphabet Σ with two special symbols <s> and </s> marking the beginning and the end of each sentence, respectively. For that purpose, we prefix our corpus WRL with $N - 1$ <s>-symbols and append $N - 1$ </s>-symbols at its end (this also simplifies the computation of the conditional probabilities, see Section 6). Fig. 2 shows an example for $N = 3$. Note that the delimiter symbols are treated in an optimised manner.

Counting is performed by applying the counting WRT $\mathcal{F}_N$ to the weighted language $\mathcal{K}$ given by the corpus:

Definition 17 (N-gram counting). *Given a WRL $\mathcal{K} : \Sigma^* \to \mathcal{R}$ representing a corpus, the N-gram counts $\mathcal{C}_N : \Sigma^* \to \mathcal{R}$ are obtained by:*

$$\mathcal{C}_N = \mathcal{F}_N[\mathcal{K}]\ .$$

a final state p (printed as a double circle) is stated after /. If the weight is omitted, it is assumed to be $\overline{1}$.

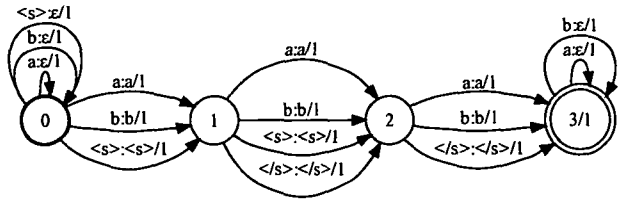


Figure 2: Transducer for counting trigrams over $\Sigma = \{a, b, <s>, </s>\}$.

We also call $\mathfrak{C}_N$ an *N-gram count WRL*. For details on the procedure and a proof of its correctness we refer the reader to [2].

The trigram counts for the example corpus (Figure 1) are shown in Figure 3 (after optimising – that is removal of ε-transitions, determinisation, and minimisation – the corresponding WFSA). Note that for the purpose of demonstrating non-robust language models first (cf. Section 6) we have chosen a corpus over $\Sigma = \{a, b, <s>, </s>\}$ which contains each meaningful trigram in Σ^N at least once resulting in an almost complete WSA.[6] Note that trigrams ending in <s> or starting with </s> cannot exist.

To get the count $C(w_1 \ldots w_N)$ associated with a specific N-gram $w_1 \ldots w_N$ we compute $[\![w_1 \ldots w_N]\!]^{\mathfrak{C}_N}$ – the weight assigned to $w_1 \ldots w_N$ by $\mathfrak{C}_N$ according to Definition 16. For example, $[\![ab</s>]\!]$ of Figure 3 is $1 \cdot 28 \cdot 0.5 \cdot 0.5 \cdot 1 = 7$.

4.3 Implementation and Complexity

The structure and therefore the size of the WFST $\mathfrak{F}_N$ corresponding to $\mathcal{F}_N$ depends on the model parameter N and the size of the underlying alphabet. Its state number $|Q|$ equals $N+1$ and the number of transitions $|E|$ is $|\Sigma|(N+2)$. Its space complexity is within $O(N|\Sigma|)$, thus the size of $\mathfrak{F}_N$ may become problematic for huge alphabets. As already suggested in [2], a solution to this problem are *lazy* automata, the states and transitions of which are constructed on-demand. Such automata are usually obtained from lazy versions of the finite-state algorithms. For example, an algorithm for the lazy composition of WRTs is presented in [28]. The drawback of such approaches is that the basic operands have to be explicitly represented.

Other approaches (among others, see [4]) try to construct automata *virtually* right from the beginning. Regularities in their structure are used to define states

[6] A (W)FSA is called *complete* with respect to an alphabet Σ if each state has outgoing transitions for each symbol $a \in \Sigma$.

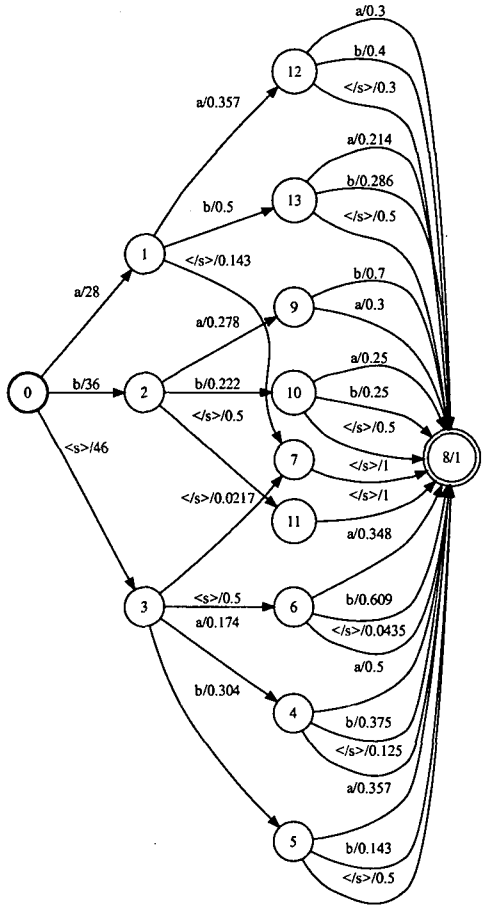


Figure 3: Trigrams in the toy corpus after optimisation.

and transitions implicitly by some calculation specification.

The simple structure of $\mathfrak{F}_N$ makes it suitable for a virtual construction: The set of states Q is simply $\bigcup_{q=0}^{N}\{q\}$ with N being the only final state. The set of transitions E has three different subsets: E_i, containing all transitions from the initial state, E_m, containing all transitions from non-initial and non-final states and E_f containing all transitions to the final state. Transitions in E_m for example

lead from state q to state $q+1$ with each symbol $a \in \Sigma$ while emitting this symbol. The formal construction of $\mathfrak{F}_N$ can be found in Definition 35 in Appendix B.

Definition 35 enables a virtual construction. Implementations of access functions to states and transitions work in O(1) time while consuming only a constant amount of memory. We have implemented this special representation of $\mathfrak{F}_N$ within the framework of [12].

Given a corpus WFSA $\mathfrak{K}$ and an N-gram counter $\mathfrak{F}_N$, counting is performed most efficiently by the following sequence of automata operations:

$$\mathfrak{C}_N = min(det(rm\text{-}\varepsilon(\pi^2(\mathfrak{K} \circ \mathfrak{F}_N)))) \; . \tag{13}$$

Since the number of N-gram paths after composition is bounded by $|\mathfrak{K}|$ and since the result is acyclic, ε-removal, determinisation (which is essentially the construction of a trie from the found N-grams), and minimisation (including weight-pushing) can be performed in $O(|\mathfrak{K}|)$ time [27, 25, 24, 13].[7]

5 Probabilisation

The next step in constructing an N-gram language model is to compute the conditional probabilities of the events according to their frequency. This is done by normalising their counts (this equation is also called *maximum likelihood estimation*, see [16]):

$$\Pr(w_i | w_{i-N+1}^{i-1}) = \frac{\mathrm{C}(w_{i-N+1}^{i-1} \cdot w_i)}{\sum_{a \in \Sigma} \mathrm{C}(w_{i-N+1}^{i-1} \cdot a)} \; . \tag{14}$$

Thus, the frequency of an N-gram is divided by the sum of the frequencies of all N-grams sharing the same $(N-1)$-gram prefix.

5.1 Conditional Probabilities

In order to normalise the N-gram counts as stated in equation (14), the weights of all N-grams sharing the same $(N-1)$-gram prefix have to be collected. Both parts of the division need to have the same language projection to guarantee that no N-grams are lost. The N-grams are therefore 'reweighted' by their corresponding collected prefix weights. This reweighting is done by a suffix expansion performed by a WRT $\mathcal{E}_N^k : \Sigma^N \times \Sigma^N \to \mathcal{R}$ which maps all N-gram suffixes of length k to each other, what effectively assigns each weight to every symbol.

Definition 18 (Suffix expansion). *Given a finite alphabet Σ and model parameters $N > 0$ and $k \leq N$, a WRT $\mathcal{E}_N^k : \Sigma^N \times \Sigma^N \to \mathcal{R}$ is defined as*

$$\forall x, y \in \Sigma^N, \; \mathcal{E}_N^k(x, y) = (ID(\Sigma^{N-k}) \cdot (\Sigma \times \Sigma)^k)\,(x, y) \; .$$

[7] $|\mathfrak{A}| = |Q_{\mathfrak{A}}| + |E_{\mathfrak{A}}|$, that is, the size of a WFSA $\mathfrak{A}$ is measured in terms of the size of its state set and its number of transitions.

The first $N-k$ steps of this transduction are an identity mapping. The following k steps create – to speak in terms of the underlying WFST – non-deterministic paths: The crossproduct of Σ with Σ results in $|\Sigma|$ transitions for every symbol $a \in \Sigma$. The corresponding transducer for $\mathcal{E}_3^1$ is shown in Figure 4.[8] By applying

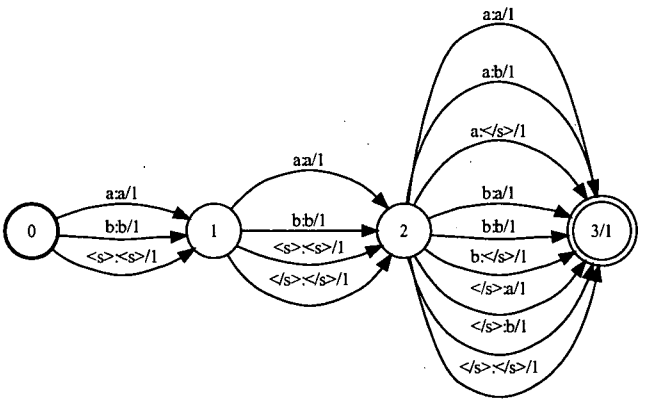


Figure 4: The unigram suffix expansion for trigrams $\mathfrak{E}_3^1$ for $\Sigma = \{a, b, \text{<s>}, \text{</s>}\}$.

$\mathcal{E}_N^1$ to the N-gram counts, the weights of all N-grams are expanded. The chosen $k = 1$ cares for the summing over the unigram suffixes and the N-grams bear the sum of the weight of the N-grams sharing the same $(N-1)$-gram prefixes as demanded by Equation (14). The extended weights are $\otimes$-negated and intersected with the N-gram counts to perform the normalisation. Given the N-gram counts $\mathcal{C}_N$ as computed in Section 4, $\mathcal{P}_N^c(\mathcal{C}_N) : \Sigma^N \to \mathcal{R}, w = w_1^N \mapsto \Pr(w_N|w_1^{N-1})$ implements this series of rational operations.

Definition 19 (Conditional N-gram probabilisation). *Given a WRL* $\mathcal{C}_N : \Sigma^N \to \mathcal{R}$, $w_1^N \mapsto \mathrm{C}(w)$, $\mathcal{P}_N^c(\mathcal{C}_N)$ *is defined as*[9]

$$\mathcal{P}_N^c(\mathcal{C}_N) = \left(\mathcal{C}_N \cap (\mathcal{E}_N^1[\mathcal{C}_N])^{-\bar{1}}\right) .$$

An example of the application of Definition 19 is shown in Figure 5.

In Figure 5, the probability of seeing a b after having seen an ab – that is, $\Pr(b|ab) = [\![abb]\!]$ – is 0.4.

[8] Again, some transitions related to the delimiters were removed for reasons of clarity.

[9] Note that the *joint N-gram probabilisation* (which reflects the joint probability of each N-gram), is computed by $\mathcal{P}_N^j(\mathcal{C}_N) = \left(\mathcal{C}_N \cap (\mathcal{E}_N^N[\mathcal{C}_N])^{-\bar{1}}\right)$. The language weight of such an probabilisation, that is $\bigoplus_{x \in \mathcal{C}_N} \mathcal{P}_N^j(\mathcal{C}_N)(x)$, equals $\bar{1}$.

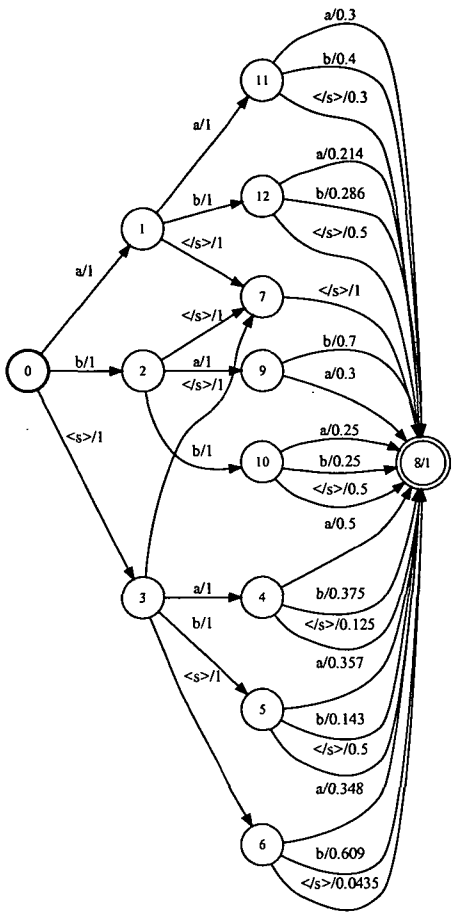


Figure 5: Conditional probabilised trigrams from the example corpus.

Lemma 1 (Correctness of conditional N-gram probabilisation). *Definition 19 computes the conditional probability of each N-gram as a special case of Equation (14) (with $i = N$):*

$$\Pr(w_N|w_1^{N-1}) = \frac{\mathrm{C}(w_1^N)}{\sum\limits_{a\in\Sigma} \mathrm{C}(w_1^{N-1} \cdot a)} \,. \tag{15}$$

Proof. See Appendix A. □

Note that an advantage of the automata/language theoretic approach is that Definition 19 computes the conditional probabilities of all found N-grams simultaneously.

5.1.1 Implementation and Complexity

The most efficient implementation of Definition 19 in terms of WFSA operations is the following:

$$\mathfrak{C}_N \cap neg^{\otimes}(min(det(\pi^2(\mathfrak{C}_N \circ \mathfrak{E}_N^1)))) \; . \tag{16}$$

A problem could arise through the constant factor associated with the alphabet size in Definition 18, since the number of transitions in a WFSA corresponding to $(\Sigma \times \Sigma)^k$ is $|\Sigma|^{2k}$. So the approach may become unfeasible in case of the big alphabet sizes commonly encountered in corpus linguistics. The composition operation $\circ$ maps every transition t in $\mathfrak{C}_N$ leading to a final state to $|\Sigma|$ transitions in the result. Since the operand of $neg^{\otimes}$ must be deterministic, all transitions resulting from suffix expansion must be (additively) combined by determinisation.

To get rid of the constant introduced by the size of the alphabet, we define a special symbol <?>, called the *default symbol* (see [5]). During intersection and composition, <?> matches every unmatched symbol labeling a transition leaving a state q. The definition of suffix expansion is then changed to the one in Definition 20:

Definition 20 (Revised suffix expansion). *Given two finite alphabets Σ and Δ and model parameters $N > 0$ and $k \leq N$, a WRT $\mathcal{E}_N^{k,\Delta} : \Sigma^N \times (\Sigma^{N-k} \cdot \Delta^k) \to \mathcal{R}$ is defined as*

$$\forall x, y \in \Sigma^N, \; \mathcal{E}_N^{k,\Delta}(x, y) = (ID(\Sigma^{N-k}) \cdot (\Sigma \times \Delta)^k)\,(x, y) \; .$$

Note that $\mathcal{E}_N^k$ is a special case of Definition 20. The special suffix expansion using <?> is then $\mathcal{E}_N^{k,\{<?>\}}$.

To reflect the special semantics of <?>, the implementations of $\cap$ and $\circ$ are changed to $\cap^{<?>}$ and $\circ^{<?>}$, respectively. Equation (16) becomes

$$\mathfrak{C}_N \cap^{<?>} neg^{\otimes}(min(det(\pi^2(\mathfrak{C}_N \circ^{<?>} \mathfrak{E}_N^1)))) \; . \tag{17}$$

The complexity of the suffix expansion, projection, determinisation and minimisation is then in $O(|\mathfrak{C}_N|)$. If we assume that $\mathfrak{C}_N$ is deterministic, the complexity of the final intersection step is also in $O(|\mathfrak{C}_N|)$, since both operands contain exactly the same N-grams (they have the same language projection), thus are isomorphic.

The possible types of symbols in a (W)FSA may be cross-classified according to Table 2. Following Table 2, the default symbol <?> can be seen as a conditionally interpreted input consuming symbol. We will need its non-consuming counterpart, the *failure transition symbol* ϕ (see [1]) in Section 7 to create robust back-off language models.

	+consuming	−consuming
+conditional	<?>	ϕ
−conditional	$a \in \Sigma$	ε

Table 2: A cross-classification of symbols labeling transitions in an FSA

In parallel to the counting WRT, it is possible to define a calculation for $\mathfrak{C}_N^{k,\Delta}$ which enables its virtual construction. The calculation is given in Definition 36 (see Appendix B).

We move to the creation of non-robust language models.

6 Creating Non-Robust Language Models

The result of the counting and the normalisation procedure $\mathcal{P}_N^c$ is a weighted language $\Sigma^N \to \mathcal{R}$. It assigns the conditional probability $\Pr(w_i|w_{i-N+1}^{i-1})$ to every N-gram in the corpus. A maximum likelihood model is characterised by the following equation:

$$\Pr(w_1^m) = \prod_{i=1}^{m} \Pr(w_i|w_{i-N+1}^{i-1}) \,. \tag{18}$$

It is a weighted language $\Sigma^* \to \mathcal{R}$. Therefore, $\mathcal{P}_N^c$ has to be transformed to accept sequences of any length. Simply taking its closure is not sufficient, since the result would be a mapping from $(\Sigma^N)^* \to \mathcal{R}$: every N-gram could be followed by any other N-gram, every input symbol would have to be processed N times (as illustrated in example 1) and only strings with a length equal to a multiple of N would be in its domain.

Example 1 (Illustration of the necessary bigram overlapping).

Given input		a		b		c
		w_1		w_2		w_3
$\Pr(w_1^3)$	=	$\Pr(a)$	·	$\Pr(b\,\|\,a)$	·	$\Pr(c\,\|\,b)$
To process (overlap)		a		ab		bc

To correctly reflect Equation (18), N-grams need to be overlapped in a way such that every $(N-1)$-gram suffix is simultaneously treated as an $(N-1)$-gram prefix. In order to achieve this, a specialisation of the concatenation operation called *overlapping* or *domino concatenation* is introduced.

Definition 21 (Domino (Overlapping) Concatenation). *The overlapping concatenation of two WRTs* $\mathcal{S} : \Sigma^* \times \Delta^* \to \mathcal{R}$ *and* $\mathcal{Q} : \Sigma^* \times \Delta^* \to \mathcal{R}$ *– denoted by* $\mathcal{S} \cdot_N \mathcal{Q}$ *– is a mapping* $\Sigma^* \times \Delta^* \to \mathcal{R}$ *defined by*

$$\forall x \in \Sigma^*, \forall y \in \Delta^*,\ (\mathcal{S} \cdot_N \mathcal{Q})(x,y) = \bigoplus_{x = u \cdot v_1^{N-1} \cdot w, y = st} \mathcal{S}(u \cdot v_1^{N-1}, s) \otimes \mathcal{Q}(v_1^{N-1} \cdot w, t) \,.$$

The $\cdot_N$ operator is rational, as long as N is a constant.

6.1 N-Gram Models as WRLs

With the overlapping concatenation at hand, it is possible to use the closure of the conditional probability distribution as the basis of the N-gram model. It is used to filter non-overlapping sequences of N-grams. A transduction $\mathcal{D}_N : (\Sigma^N)^* \times \Sigma^* \to \mathcal{R}$ is defined which uses $\cdot_N$ in this way. To avoid their multiple processing, $\mathcal{D}_N$ deletes overlapping prefixes by simply omitting them from its output.

Definition 22 (N-gram Concatenator). *Given a finite alphabet Σ, the N-gram concatenator is a WRT $\mathcal{D}_N : (\Sigma^N)^* \times \Sigma^* \to \mathcal{R}$, defined as*

$$\forall x, y \in \Sigma^*,\ \mathcal{D}_N(x,y) = \left(\Sigma^N \cup \left(\Sigma^N \cdot_N \overset{\infty}{\underset{i=0}{\textcircled{\cdot_N}}} \left(\bigcup_{w_1^N \in \Sigma^N} \{(w_1^N, w_N)\}\right)\right)\right)(x,y)\ .$$

Fig. 6 shows a trigram concatenator for $\Sigma = \{a, b\}$. Note that the N-gram con-

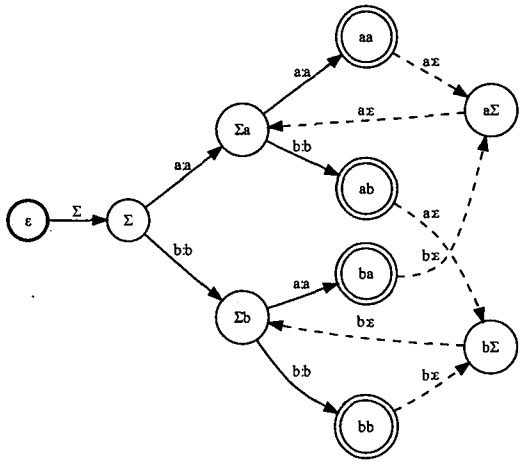


Figure 6: Trigram concatenator for $\Sigma = \{a, b\}$. States are labeled with their histories. The dashed transitions correspond to the overlaps.

catenator factors out the structure of an N-gram model (cf. [14], p.83) and makes it available to the algebraic formalisation independently from the corpus under consideration.

To handle the special cases for $1 \leq M < N$ in Equation (18) uniformly, we prefix our input sentence with $N-1$ <s>-symbols marking the sentence begin. Additionally, we postfix it with the same number of </s>-symbols marking its end, in order to guarantee that our language model seen as a WFSA has a unique

final state (which is reached after reading the last </s>-symbol). For the model's structure, this means that only those N-grams starting with $(\text{<s>})^{N-1}$ and those ending in $(\text{</s>})^{N-1}$ may be accepted in the beginning and at the end, respectively. To reflect this, we *unfold* the closure of the conditional probabilities $\mathcal{P}_N^c$ by intersecting it with the WRL $\mathcal{U}_N$.

Definition 23 (Unfolding N-grams). *Let Σ be an alphabet and N the model parameter. $\mathcal{U}_N : \Sigma^* \to \mathcal{R}$ is defined as:*

$$\forall x \in (\Sigma^N)^*,\ \mathcal{U}_N(x) = \left(\{\text{<s>}^{N-1}\} \cdot \Sigma \cdot (\Sigma^N)^* \cdot \Sigma \cdot \{\text{</s>}^{N-1}\}\right)(x) \ .$$

Definition 24 applies the N-gram concatenator $\mathcal{D}_N$ to the intersection of the closure of the probabilised N-grams and the unfolding WRL.

Definition 24 (Non-robust language models). *Let $\mathcal{C}_N$ be an N-gram count WRL as defined in Definition 17, such that $\mathcal{C}_N(x) \neq 0$, $\forall x \in \Sigma^N$. The* non-robust language model $\mathcal{M}_N(\mathcal{C}_N)$ *is a weighted rational transduction $\Sigma^* \to \mathcal{P}, x \in \Sigma^+ \mapsto \Pr(x)$*

$$\mathcal{M}_N(\mathcal{C}_N) = \mathcal{D}_N[(\mathcal{P}_N^c(\mathcal{C}_N))^* \cap \mathcal{U}_N] \ .$$

Note that for the following theorem, we make the assumption that our input corpora are complete, that is, they contain every possible N-gram $w \in \Sigma^N$. We will relax this condition in Section 7.

Theorem 1 (Adequacy of Definition 24). *$\mathcal{M}_N(\mathcal{C}_N)(w)$ correctly computes the decomposed conditional probability of Equation (18) for each delimited input string w.*

Proof. The proof is a special case (the two cases 1a) of the proof of Theorem 2 (cf. Appendix A). □

There is a relation between automata representing N-gram models and *de Bruijn graphs* [7]: A de Bruijn graph is a directed graph which represents the overlaps of sequences of a certain length n given a finite alphabet Σ. Each length n sequence of symbols in Σ is represented as a vertex in the graph. Let q denote the vertex for a sequence w_i^{i+n-1}, then q has a single edge for each symbol $a \in \Sigma$ connecting it to the vertex r representing $w_{i+1}^{i+n-1} \cdot a$. Thus, the structure of de Bruijn graphs is comparable to that of N-gram models over complete corpora.

6.2 Implementation and Complexity

Again, combining the WFSA for $\mathcal{P}_N^c$ and the WFST for $\mathcal{D}_N$ is basically application followed by optimisation:

$$\mathfrak{M}_N = rm\text{-}\varepsilon\left(\pi^2\left(((\mathfrak{P}_N^c)^* \cap \mathfrak{U}_N) \circ \mathfrak{D}_N\right)\right) \ . \tag{19}$$

If $(\mathfrak{P}_N^c)^* \cap \mathfrak{U}_N$ is deterministic and since $\mathfrak{D}_N$ is input deterministic by definition, their composition will be input deterministic too. After taking the 2^{nd} projection,

the ε-transitions resulting from the overlaps have to be removed. Although the result is a cyclic WFSA, its (unconnected) ε-subgraph will be acyclic, in fact, we will find a number of unconnected ε-chains of length $N-1$. These ε-transitions can be removed in linear time with respect to the size of the result of the application [23], which in turn is bounded by the size of $\mathfrak{P}_N^c$. Thus, the time complexity of Equation (19) is in $O(|\mathfrak{P}_N^c|)$.

Even though $\mathfrak{D}_N$ depends only on the (constant) alphabet Σ and the model constant N, its space complexity is in $O(\Sigma^{N-1})$, since $\mathfrak{D}_N$ has to keep track of the different histories of length $N-1$ to ensure correct overlaps. So, a naive implementation runs into difficulties even with moderate alphabet sizes. But we can do better if we exploit the regular structure of $\mathfrak{D}_N$ and replace actual states and transitions by functions computing them on demand. The trigram concatenator of Figure 6 is shown slightly modified in Figure 7. Labels of states have been replaced by state numbers and two additional states are introduced to simplify the virtual construction. In addition, we assume a bijective function `idx` : $\Sigma \to \mathbb{N}$ mapping each alphabet symbol to a unique index r, $0 \leq r < |\Sigma|$. The labels of the transitions are replaced by their corresponding indices. Ignoring state 0, the first part of the

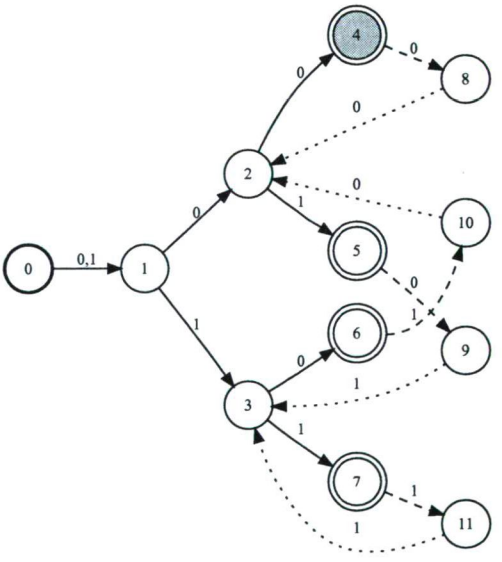


Figure 7: Trigram concatenator for $\Sigma = \{a, b\}$. States are labeled with numbers.

automaton shown in Figure 7 can be seen as a binary tree with root 1, yield 4...7 and a consecutive labeling. The successor of a state q given an alphabet symbol a

can be calculated by $q * |\Sigma| + \mathtt{idx}(a) - (|\Sigma| - 2)$ in the general case.

Example 2. Consider state 3 and symbol b with $\mathtt{idx}(\mathrm{b}) = 1$ in Figure 7. The correct destination state of the transition is state 7. Thus,

$$7 = 3 * 2 + 1 - (2 - 2).$$

The transitions within the tree part are denoted by E_t.

Transitions from states greater or equal than the first state of the yield q_y (state 4 in Figure 7) perform the overlap.

Definition 25 (Calculation of q_y). *Given a finite alphabet Σ and a model parameter N, the state q_y is calculated as follows:*

$$q_y = \frac{|\Sigma|^{N-1} + (|\Sigma| - 2)}{|\Sigma| - 1}.$$

q_y is used to identify the states which do not allow branching. The transitions leaving those states are divided into the overlap transitions E_o and the loop transitions E_l. The computation of their destinations is simple, but one has to take care of the fact that only one symbol may be processed.

The complete calculation specification which enables a virtual construction of $\mathfrak{D}_N$ is given in Definition 37 in Appendix B. The virtual construction of $\mathfrak{U}_N$ is straightforward.

The next section focuses on *robust language models.*

7 Robust Language Models

Up to this point, the achieved models are only robust when based on corpora containing all possible N-grams which is an unrealistic assumption. As described in Section 2.2, smoothing methods have to be applied in order to solve this problem. Back-off smoothing can be described as 'relying on the highest order distribution which is available'. The following figure illustrates this behavior on the automata level (taken from [2]):

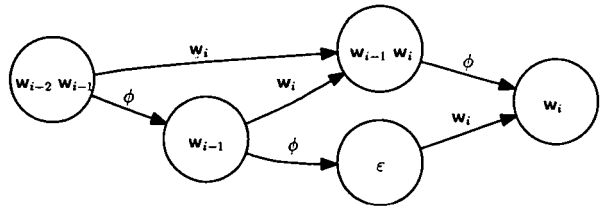


Figure 8: A trigram back-off model represented as a schematic FSA.

As suggested in [2], in those cases where – given a specific history – no transition for the next word w_i is available, a *failure transition* (marked by ϕ) to the nearest

shorter history is traversed and w_i is processed if possible. If not, the history is shortened again until the history-less state is reached. Language models as achieved in Section 6 have to be extended to include such failure transitions to the lower ordered distributions. Following Section 2.2, it is necessary to apply a discounting algorithm in order to free probability mass for them.

For computing the probability of a (delimited) string w_1^m in a back-off model, we use the Markov probability decomposition as in Equation (18), but replace Pr with the back-off probability $\hat{\Pr}$:

$$\Pr(w_1^m) = \prod_{i=1}^{m} \hat{\Pr}(w_i | w_{i-N+1}^{i-1}) \ . \tag{20}$$

7.1 Discounting

From the many existing discounting approaches, it is especially Witten-Bell discounting which is suited for modifying N-gram counts in a finite-state algebraic manner. The calculations for the discounted frequencies as well as for the freed frequency mass were given above in equations (5) and (6).

As explained above, Witten-Bell discounting uses the number of observed types following a history to estimate the probability of previously unseen events. Frequencies are discounted in relation to this number. Given a representation of N-gram counts, the number of types for each history can be computed with the help of the language projection (Definition 12) and the suffix expansion operator $\mathcal{E}_N^k$ (Definition 18). The idea is to first map all N-gram counts to 1 and then sum over the 1-gram suffixes.

Definition 26 (Witten-Bell Type Number). *Given a WRL $\mathcal{L} : \Sigma^N \to \mathcal{R}$, a WRL $\mathcal{T}_N : \Sigma^N \to \mathcal{R}$ is defined as follows:*

$$\mathcal{T}_N(\mathcal{L}) = \mathcal{E}_N^1[\pi^L(\mathcal{L})] \ .$$

$\mathcal{T}_N$ directly corresponds to function T from Definition (1).

Lemma 2 (Correspondence of T and $\mathcal{T}_N$). *Given a WRL $\mathcal{L} : \Sigma^N \to \mathcal{R}$, $\forall w_1^N \in \Sigma^N : \mathcal{T}_N(\mathcal{L})(w_1^N) = \mathrm{T}(w_1^N)$.*

Proof. See Appendix A. □

Definition 27 defines the analogon to N of Definition 2.

Definition 27 (Witten-Bell Token Number). *Given a WRL $\mathcal{L} : \Sigma^N \to \mathcal{R}$, a WRL $\mathcal{N}_N : \Sigma^N \to \mathcal{R}$ is defined as follows:*

$$\mathcal{N}_N(\mathcal{L}) = \mathcal{E}_N^1[\mathcal{L}] \ .$$

Lemma 3 (Correspondence of N and $\mathcal{N}_N$). *Given a WRL $\mathcal{L} : \Sigma^N \to \mathcal{R}$, $\forall w_1^N \in \Sigma^N : \mathcal{N}_N(\mathcal{L})(w_1^N) = \mathrm{N}(w_1^N)$.*

Proof. The proof is analogous to the proof of Lemma 2. □

The nominator of Equation (5) (which is at the same time the first summand of the denominator) has been used for obtaining conditional probabilities before (Section 5). Thus, everything needed for Witten-Bell discounting is at hand: we reconstruct Equation (5) using corresponding operations on WRLs. To reflect the N-gram discounting process, we actually operate on $\mathcal{C}_N$.

Definition 28 (Witten-Bell Discounting). *Given a WRL* $\mathcal{L} : \Sigma^N \to \mathcal{R}$, *we define* $\mathcal{W}_N^D(\mathcal{L}) : \Sigma^N \to \mathcal{R}, w \in \Sigma^N \mapsto \tilde{\mathrm{C}}(w)$ *as*

$$\mathcal{W}_N^D(\mathcal{L}) = \mathcal{L} \cap (\mathcal{N}_N(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\bar{1}}) ,$$

and $\mathcal{W}_N^R(\mathcal{L}) : \Sigma^N \to \mathcal{R}, w \in \Sigma^N \mapsto \mathrm{C}(w) - \tilde{\mathrm{C}}(w)$ *as*

$$\mathcal{W}_N^R(\mathcal{L}) = \mathcal{L} \cap (\mathcal{T}_N(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\bar{1}}) .$$

The second part of Definition 28 computes the freed frequency mass by reformulating Equation (6).

Again, we make use of the fact that the real semiring $\mathcal{R}$ is closed under multiplicative inverses to show that Definition 28 corresponds to the Witten-Bell discounted frequencies (resp. the freed frequency mass).

Lemma 4 (Reconstruction of Witten-Bell Discounting). *Given an N-gram count WRL* $\mathcal{C}_N : \Sigma^N \to \mathcal{R}$, $w_1^N \mapsto \mathrm{C}(w_1^N)$, $\mathcal{W}_N^D(\mathcal{C}_N)(w_1^N)$ *maps an N-gram to its Witten-Bell discounted frequency* $\tilde{\mathrm{C}}(w_1^N)$.

Proof. See Appendix A. □

The following equivalence holds:

Lemma 5 (Witten-Bell Decomposition). *Given an N-gram count WRL* $\mathcal{L} : \Sigma^N \to \mathcal{R}$, $\mathcal{W}_N^D(\mathcal{L}) \cup \mathcal{W}_N^R(\mathcal{L}) = \mathcal{L}$.

Proof. See Appendix A. □

An example of the discounting process is shown in Figure 9. Both parts of the Witten-Bell decomposition are used for reconstructing the back-off strategy as explained in the next section.

7.2 Back-off

The previously reserved frequency mass now has to be reallocated to the lower ordered distributions which need to be discounted as well (except the unigram distribution terminating the recursion). All involved distributions are then combined in a special representation to which the *robust overlapping concatenation* operator is applied.

The first step is to transform the adjusted frequencies into conditional probabilities. In principle, the procedure from Section 5 can be used with the difference that

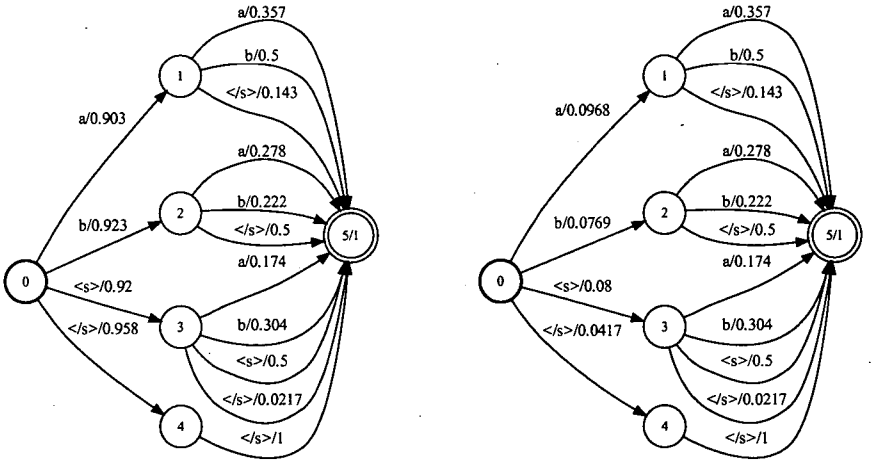


Figure 9: Witten-Bell decomposition for the bigrams of the corpus. The WFSA on the left is the discounted WFSA. Both WFSAs are already probabilised after Definition 29.

both have to be normalised in relation to the original counts instead of normalising them in relation to themselves. $\mathcal{P}_N^c$ is therefore modified to use the discounted frequencies (resp. the discounts, indicated by a second superscript) as the first argument of the integrated intersection operation.

Definition 29 (Witten-Bell Discounted Probabilities). *Let $\mathcal{L}$ denote an N-gram count WRL $\Sigma^N \to \mathcal{R}$, then $\mathcal{P}_N^{c,D} : \Sigma^N \to \mathcal{R}$ is defined as*

$$\mathcal{P}_N^{c,D}(\mathcal{L}) = \mathcal{W}_N^D(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}))^{-\overline{1}},$$

and $\mathcal{P}_N^{c,R} : \Sigma^N \to \mathcal{R}$ is defined as

$$\mathcal{P}_N^{c,R}(\mathcal{L}) = \mathcal{W}_N^R(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}))^{-\overline{1}}.$$

$\mathcal{P}_N^{c,D}$ and $\mathcal{P}_N^{c,R}$ denote the Witten-Bell discounted probabilities and the freed probability mass of the N-grams when applied to $\mathcal{C}_N$, respectively. Note that the union of $\mathcal{P}_N^{c,D}$ and $\mathcal{P}_N^{c,R}$ yields $\mathcal{P}_N^c$.

Lemma 6 (Witten-Bell Discounted Probabilities). *Given $\mathcal{C}_N : \Sigma^N \to \mathcal{R}, w = w_1^N \in \Sigma^N \mapsto \mathrm{C}(w)$, $\mathcal{P}_N^{c,D}(\mathcal{C}_N)(w)$ and $\mathcal{P}_N^{c,R}(\mathcal{C}_N)(w)$ compute $\tilde{\Pr}(w_N|w_1^{N-1})$ and $\breve{\Pr}(w_N|w_1^{N-1})$, the Witten-Bell discounted probabilities and the freed probability mass, respectively.*

Proof. Lemma 6 results from Lemma 1 and Lemma 4. □

Lemma 7 (Union of $\mathcal{P}_N^{c,D}$ and $\mathcal{P}_N^{c,R}$). *Let $\mathcal{L}$ denote an N-gram count WRL $\Sigma^N \to \mathcal{R}$:*

$$\mathcal{P}_N^{c,D}(\mathcal{L}) \cup \mathcal{P}_N^{c,R}(\mathcal{L}) = \mathcal{P}_N^{c}(\mathcal{L}) \ .$$

Proof. See Appendix A. □

7.2.1 The Unified Distribution

To create a model which contains all $N \ldots 1$-gram distributions, these have to be combined in some way. The aim is to enable the application of an overlapping filter - as in the non-back-off case - to the closure of the combination $\mathcal{Y}_N$ which therefore must, according to Equation (7), meet some requirements:

1. The single distributions must be discriminated from each other, since exactly one may account for a single event.
2. The single distributions must be ordered in a way that the back-off strategy is reflected.
3. The discounting factors $\alpha()$ of Equation (7) are context-dependent. They have to be assigned correctly.

The first point is realised by prefixing each M-gram distribution with $N - M$ α-symbols. Hence, their difference and hierarchy originates in the number of αs preceding them. α is a special symbol which is not part of Σ. It has no special semantics, is processed as any other symbol and will be deleted later. To comply with the third point, an α is appended to every $(M-1)$-gram prefix $(1 < M \leq N)$. This α will be identified with the back-off weight of the prefix it is attached to. We define the unified distribution $\mathcal{Y}_N$.

Definition 30 (Unified Distribution $\mathcal{Y}_N$). *Given a WRL $\mathcal{L} : \Sigma^* \to \mathcal{R}$ representing a corpus, the combined representation of all $1 \ldots N$-gram distributions $\mathcal{Y}_N(\mathcal{L}) : \Sigma^N \to \mathcal{R}$ is defined as:*

$$\mathcal{Y}_N(\mathcal{L}) = \alpha^{N-1} \cdot \mathcal{P}_1^c(\mathcal{F}_1[\mathcal{L}]) \cup \bigcup_{M=2}^{N} \left(\alpha^{N-M} \cdot \left(\mathcal{P}_M^{c,D}(\mathcal{F}_M[\mathcal{L}]) \cup \mathcal{E}_M^{1,\{\alpha\}}[\mathcal{P}_M^{c,R}(\mathcal{F}_M[\mathcal{L}])] \right) \right) .$$

The base part of $\mathcal{Y}_N(\mathcal{L})$ is defined by the unigram distribution $\mathcal{P}_1^c(\mathcal{F}_1[\mathcal{L}])$ which is prefixed with $N-1$ α-symbols. Note that in the case of unigrams, conditional and joint distributions are the same. The other part of the unified distribution contains for every M (with $1 < M \leq N$) a sublanguage which is the union of two weighted subsets: first the discounted M-gram probability distribution $\mathcal{P}_M^{c,D}(\mathcal{F}_M[\mathcal{L}])$ and second the residual probability mass $\mathcal{P}_M^{c,R}(\mathcal{F}_M[\mathcal{L}])$. For the latter, the suffix expansion WRT $\mathcal{E}_M^{1,\{\alpha\}}$ ensures that it consists of words $w_1 \ldots w_{M-1} \cdot \alpha$ whose associated weight corresponds to the $\alpha(w_1^{M-1})$-value in Equation (7) and which is computed

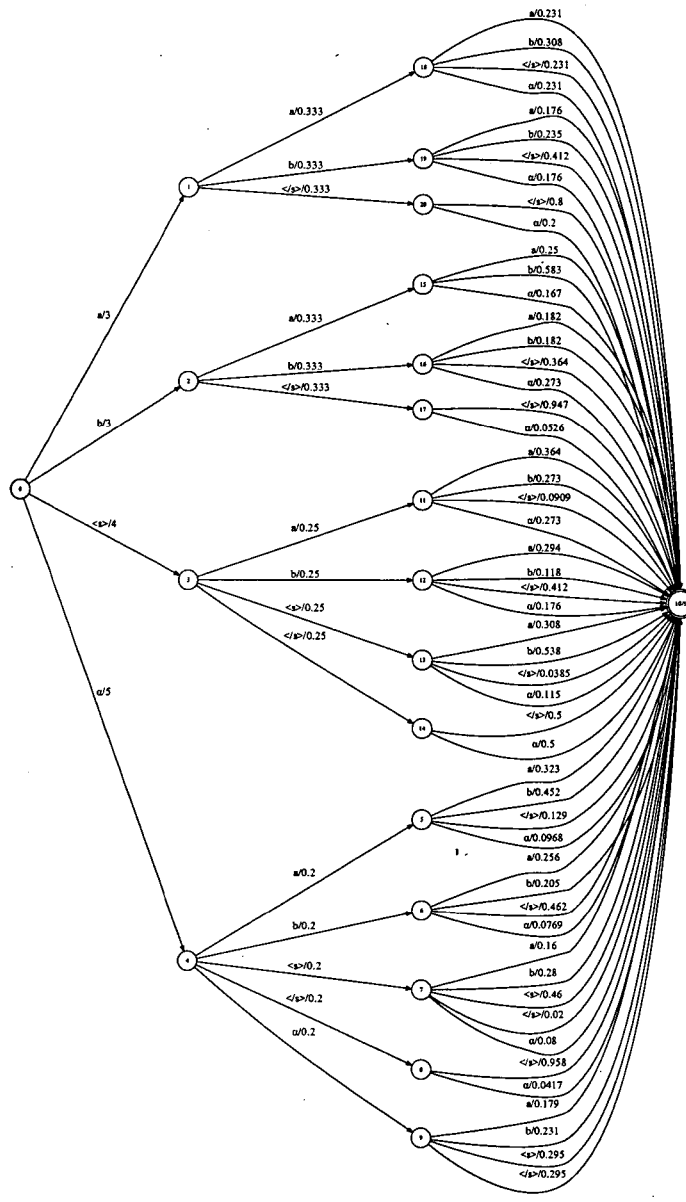


Figure 10: Unified distribution containing all $\{1, 2, 3\}$-gram subdistributions.

by the smoothing method. Note that the strings in $\mathcal{Y}_N(\mathcal{L})$ are by definition all of length N.

Fig. 10 shows the unified distribution for the trigrams of the example corpus.

Lemma 8 ($\mathcal{Y}_N$ defines a conditional probability distribution over $(\Sigma \cup \{\alpha\})^N$).

Proof. All strings in $\mathcal{Y}_N$ are of length N and are either of the form $\alpha^{N-1}\Sigma$ (unigram case) or of the form $\alpha^{N-M}\Sigma^{M-1}(\alpha|\Sigma)$ (for $1 < M \leq N$) and originate from a single subset in Definition 30 since all those subsets are mutually disjoint. In the unigram case, for each symbol $a \in \mathcal{P}_1^c(\mathcal{F}_1[\mathcal{L}])$, $\alpha^{N-1}\mathcal{P}_1^c(\mathcal{F}_1[\mathcal{L}])$ is associated with the conditional probability $\Pr(a|\alpha^{N-1})$, since $\mathcal{P}_1^c(\mathcal{F}_1[\mathcal{L}])$ is a probability distribution by construction. By Lemma 7, the union of $\mathcal{P}_M^{c,D}$ and $\mathcal{P}_M^{c,R}$ gives a conditional probability distribution over $(\Sigma \cup \{\alpha\})^M$. Prefixing it with $N - M$ αs results in a conditional probability distribution over $(\Sigma \cup \{\alpha\})^N$. □

7.2.2 Back-off Navigation

Concerning the second point in the enumeration above, the possible sequences of M-grams according to Equation (7) have to be taken into account.

Example 3. Consider the trigram case and the input *abcde*, $c|ab$ has been processed, thus $d|bc$ is to be read next. If the trigram *bcd* and the bigram *cd* are not available we back-off successively to $d|c$ and to d. Now that d has been processed, e comes next. Since we already know that *cd* does not exist, concatenating $e|cd$ can not be correct. The correct continuation is $e|d$, the second case in Equation (9). This motivates why the w_i-transition from the ε-state in Figure 8 first traverses a bigram state before eventually going back to the trigram level.

Simply using the closure of $\mathcal{Y}_N$ as the input of the N-gram concatenator is thus not correct. Instead, we define a WRT called *back-off navigator* which ensures that incorrect sequences of M-grams are filtered from $(\mathcal{Y}_N)^*$.

Definition 31 (Back-off Navigator). *A WRL $\mathcal{B}_N : ((\Sigma \cup \{\alpha\})^N)^* \to \mathcal{R}$ is defined for a finite alphabet Σ and the model parameter N as follows:*

$$\mathcal{B}_N = (\Sigma^N)^* \cup \mathcal{B}_{N-1,N} \ .$$

The back-off part $\mathcal{B}_{M,N}$ (with $0 \leq M < N$) is recursively defined in the following way:

$$\mathcal{B}_{M,N} = \begin{cases} \{\varepsilon\} & \text{if } M = 0 \\ \left(\Sigma^M \cdot \{\alpha \cdot \alpha^{N-M}\} \cdot \mathcal{B}_{M-1,N} \cdot \Sigma^M \cdot \{\alpha^{N-M-1}\}\right)^* & \text{if } M > 0 \ . \end{cases}$$

$\mathcal{B}_{M,N}$ accounts for the impossibility of recognizing a symbol in the $M+1$-subdistribution of an N-gram model ($0 < M < N$). This failure – indicated by α – may happen after having read M symbols. We then enter the nearest subdistribution which we find in $(\mathcal{Y}_N)^*$ after reading an α-prefix of length $N - M$.

Here, we may successfully process the M-gram or recursively back-off to the lower ordered distribution. In both cases – motivated by Example 3 and defined in Equation (9) – we continue processing in the nearest superdistribution which we find in $(\mathcal{Y}_N)^*$ after reading a prefix of $N - M - 1$ αs. Note that the length of all strings in $\mathcal{B}_N$ is a multiple of N.

Fig. 11 shows the trigram navigator implementing the back-off strategy for $N = 3$. Since we prefix the input to the model as well as the sentences of the training corpus with $N - 1$ delimiter symbols, failure can only occur after reading $N - 1$ symbols, because every suffix of an N-gram of length $N - 1$ also acts as a prefix of an N-gram (this can be easily shown by induction). This motivates why the back-off navigator in Figure 11 has α transitions only in state 2 (back-off from trigrams to bigrams) and state 5 (back-off to the unigrams). The remaining α-transitions serve to navigate to the nearest sub- (states 3, 6, 7) or superdistribution (state 9).

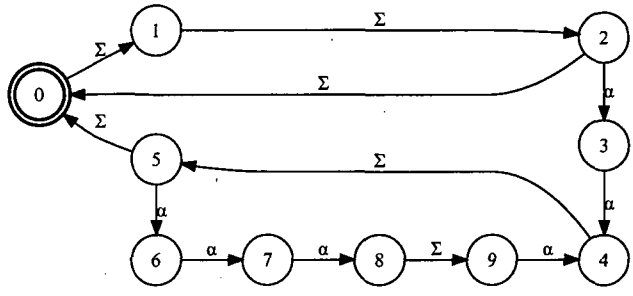


Figure 11: Back-off Navigator $\mathfrak{B}_3$.

Lemma 9 (Backoff-αs). *Let $\mathcal{P}_M^{c,R}$ $(1 < M \leq N)$ be as defined in Definition 29. For each string w_1^M, $(\mathcal{E}_M^{1,\{\alpha\}}[\mathcal{P}_M^{c,R}])(w_1^{M-1}\alpha)$ is equal to $\alpha(w_1^{M-1})$ in Equation (7).*

Proof. As defined in Equation (9), $\alpha(w_1^{M-1})$ is the residual probability mass computed by the discounting method for history w_1^{M-1}. By Lemma 6, $\mathcal{P}_M^{c,R}$ contains exactly that probability mass for all M-grams. By definition of application, $(\mathcal{E}_M^{1,\{\alpha\}}[\mathcal{P}_M^{c,R}])(w_1^{M-1}\alpha)$ maps the sum of all conditional probabilities of all strings $w_1^{M-1}a$ for $a \in \Sigma$ to $w_1^{M-1}\alpha$. □

7.2.3 Robust Overlapping Concatenation

The overlapping concatenation $\cdot_N$ is the basis for the operator $\mathcal{D}_N$ which filters sequences of non-overlapping N-grams from the closure of all N-grams $(\Sigma^N)^*$. In

parallel, a *robust* overlapping concatenation $\cdot_N^{\phi}$ is defined which allows the shortening and extension of histories during overlapping.

Definition 32 (Robust Overlapping Concatenation). *The robust overlapping concatenation $\mathcal{S} \cdot_N^{\alpha} \mathcal{Q}$ of two weighted transductions $\mathcal{S}$ and $\mathcal{Q}$ is a mapping $(\Sigma \cup \{\alpha\})^* \times (\Delta \cup \{\alpha\})^* \to \mathcal{R}$ defined by*

$$\forall x \in \Sigma^*, \forall y \in \Delta^*,\ (\mathcal{S} \cdot_N^{\alpha} \mathcal{Q})(x,y) = \mathcal{S}(x,y) \cdot_N \mathcal{Q}(x,y) \cup \bigoplus_{x=u\cdot v_1^{N-2}\cdot\alpha\cdot w, y=st} \bigcup_{i=1}^{N-1} \mathcal{S}(u \cdot v_1^{N-2} \cdot \alpha, s) \otimes \mathcal{Q}(\alpha^i \cdot v_i^{N-2} \cdot w, t)\ .$$

$\cdot_N^{\alpha}$ successively increases the number of αs to be processed while shortening the N-gram history v_1^{N-2}.

Example 4. In the trigram case, Definition 32 boils down to the following cases for input $abc(d)$:[10]

$a \cdot bc$	$\cdot_N$	$bc \cdot d$	Normal, non-failure case
$a \cdot b\alpha$	$\cdot_N^{\alpha}$	$\alpha b \cdot c$	Processing in the 2-grams by shortening the history to b
$\alpha \cdot b\alpha$	$\cdot_N^{\alpha}$	$\alpha\alpha \cdot c$	Processing in the 1-grams by shortening the history to ε
$\alpha \cdot \alpha c$	$\cdot_N$	$\alpha c \cdot d$	1-grams $\to$ 2-grams

Cases 2 and 3 in Example 4 are distinguished from the others by the failure-indicating α at the last position of the first trigram. Note that the last case is handled by the standard overlapping mechanism if α is treated as a normal symbol in Σ.

Now, everything is prepared to define the WRT which repeatedly applies $\cdot_N^{\alpha}$ to an input string. The αs which trigger the shortening of the histories in Definition 32 are introduced by occurrences of failure symbols ϕ in the input string.

Definition 33 (Robust N-gram Concatenator $\mathcal{D}_N^{\phi}$). *Let $\mathcal{D}_N^{\alpha}$ be as in Definition 22 with $(\Sigma \cup \{\alpha\})$ in place of Σ and $\cdot_N^{\alpha}$ instead of $\cdot_N$. $\mathcal{D}_N^{\phi}$ is a mapping $(\Sigma \cup \{\alpha\})^* \times (\Sigma \cup \{\phi\})^* \to \mathcal{R}$ defined by*

$$\mathcal{D}_N^{\phi} = \mathcal{D}_N^{\alpha} \circ (ID(\Sigma \setminus \{\alpha\}) \cup (\{\alpha\} \times \{\phi\}))^*\ .$$

Note that $\mathcal{D}_N^{\alpha}$ outputs – as before – only the last symbol of each N-gram, which may be α in the failure case (cf. Definition 22). $\mathcal{D}_N^{\phi}$ then simply replaces this occurrence of α by ϕ. Observe furthermore that Definition 33 is over-general, since it admits more αs than necessary. This over-generality is harmless since the sequences of αs and Σs are further constrained by the back-off navigator $\mathcal{B}_N$ (see Definition 34).

Fig. 12 shows the robust version of the trigram concatenator of Figure 6. Dashed transitions correspond to backing-off to the lower bigram and unigram distributions. Note that the actual implementation of $\mathcal{D}_N^{\phi}$ (see Figure 12) uses a weaker equiv-

[10] These cases are also the base of the proof of Theorem 2 (cf. Section 7.3).

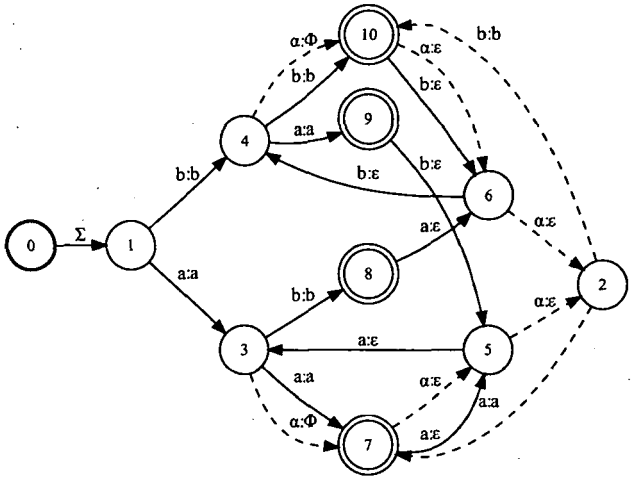


Figure 12: Robust trigram concatenator for $\Sigma = \{a, b\}$. The dashed transitions account for the back-off cases.

alence relation with respect to the states' *right relation.*[11] The implementation merges some non-equivalent states to allow for a compact representation of $\mathcal{D}_N^\phi$ which only differs minimally from the non-robust counterpart (following the back-off scheme, we would have to split for example state 10 in Figure 12 into two states to distinguish between the two possible continuations after having failed with the last input symbol b or successfully processed it. The concatenator in Figure 12 thus accepts for example the sequence $bbb\alpha ab$ which is not admissible after the back-off scheme in Figure 11). Again, this coarsening is harmless because of the filter $\mathcal{B}_N$.

7.3 Putting It All Together

The back-off language model is obtained by applying $\mathcal{B}_N$, $\mathcal{U}_N^\alpha$ and $\mathcal{D}_N^\phi$ to the unified distribution.

Definition 34 (Robust language model). *Let $\mathcal{L}$ be a weighted language over Σ^*. Let $\mathcal{U}_N^\alpha$ be the N-gram unfolder of Definition 23 where $(\Sigma \cup \{\alpha\})$ is used in place of Σ. The* robust language model $\mathcal{M}_N^\phi(\mathcal{L})$ *is a WRT $\Sigma^* \to \mathcal{P}, w \in \Sigma^* \mapsto \hat{\Pr}(w)$:*

$$\mathcal{M}_N^\phi(\mathcal{L}) = \mathcal{D}_N^\phi\left[\mathcal{Y}_N(\mathcal{L})^* \cap \mathcal{U}_N^\alpha \cap \mathcal{B}_N\right] .$$

[11] The *right relation* of a state q in a WFST $\mathcal{T}$ (*right language* in the case of WFSAs) is the WRT accepted by $\mathcal{T}$ when q is taken as the start state. Two states are equivalent (and can thus be merged during minimisation), if they have the same right relation.

We assume a mechanism which introduces the special failure symbol ϕ at the "right" places in w. Since the interpretation of ϕ is procedural in nature, we delegate this task to a special WFSA intersection algorithm. Note that the length of ϕ is 0.
From a procedural viewpoint, Definition 34 works in the following way:

- Each symbol in the input – a "normal" symbol $a \in \Sigma$ or ϕ – triggers a full cycle through $\mathcal{Y}_N$. Symbols $a \in \Sigma$ are mapped to themselves while ϕ is mapped to the appropriate $\alpha()$-value of the back-off Equation (7).
- An occurrence of ϕ is found in the input w (actually, the places where ϕ can occur are constrained by $\mathcal{Y}_N$).
- ϕ is mapped to α in the upper part of the relation by Definition 33.
- This α triggers other αs to be inserted into the relation's upper part by Definition 32.
- These additional αs determine the correct subdistribution in $\mathcal{Y}_N^*$ including the determination of the correct $\alpha()$-values of Equation (7).
- In addition, these αs are subject to the filtering of the back-off navigator $\mathcal{B}_N$ which also handles the navigation to the correct superdistribution after having read a number of ϕs.

Since complete trigram models tend to be large and our focus lies on the demonstration of the back-off mechanism, we depart from our previous example. Fig. 13 shows a back-off model following Definition 34 on the basis of the corpus a|baaaa|baaaa. For a better understanding of this example, the states are labeled with their histories. Also, the two states corresponding to the initial <s>-prefix have been deleted.

Theorem 2 (Robust language model $\mathcal{M}_N^\phi$). *Given $\mathcal{M}_N^\phi(\mathcal{L})(w)$ as defined in Definition 34, $\mathcal{M}_N^\phi(\mathcal{L})(w)$ computes the correct probability for a delimited input string w after equations (20), (7) and (9).*

Proof. See Appendix A. □

7.4 Implementation and Complexity

The observations of Section 6.2 carry over to the back-off case. Of course, the intersection of $\mathcal{U}_N^\alpha$ and $\mathcal{B}_N$ in Definition 34 can be done by a virtual intersection algorithm. Due to their sizes, all three WRTs should be virtually constructed as well.

The application of the language model $\mathcal{M}_N^\phi$ to a (delimited) input string w is as usual the intersection of the trivially weighted WFSA for w with $\mathfrak{M}_N^\phi$, the WFSA corresponding to $\mathcal{M}_N^\phi$. Since $\mathfrak{M}_N^\phi$ contains transitions labeled with the special failure symbol ϕ, the normal intersection algorithm must be augmented with a mechanism which treats ϕ as a conditional ε-transition.

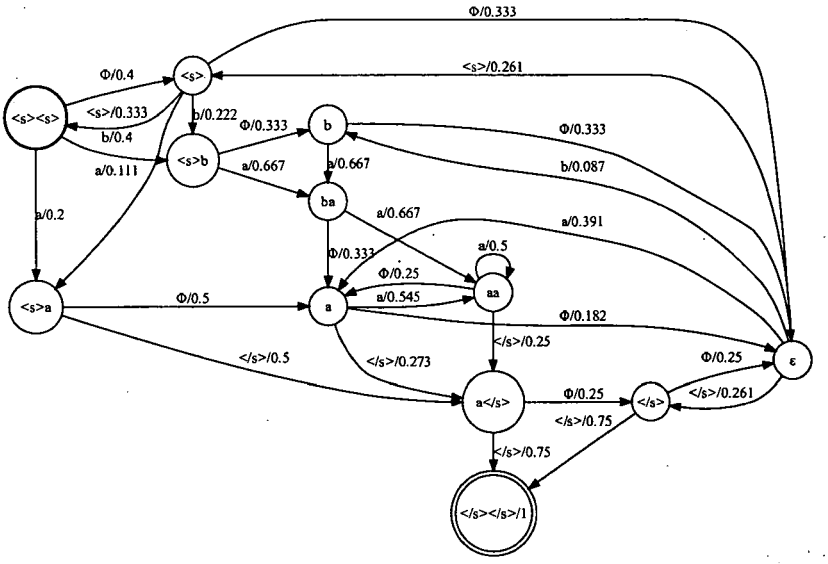


Figure 13: A back-off trigram model for the corpus $(a|baaaa|baaaa)$.

8 Conclusion and Further Work

In the previous sections, we have tried to show – to our knowledge for the first time – that the construction of language models can be defined on the formal language level alone without resorting to algorithms which manipulate the underlying WFSAs on a state level. We therefore make use of certain semiring properties, as for example divisibility, to model arithmetic statements like discounting procedures within the algebra of WRLs – an approach which may be applied to many other problems in the field of language processing.

Our formalisation is modular and can be seen as generator – $\mathcal{Y}_N^*$ – and a sequence of filters:

- The unified distribution $\mathcal{Y}_N$ accounts for the probabilities and combines discounted and residual probabilities for various values of M without caring about the back-off structure and specific N-gram histories.
- The N-gram unfolder ensures the macro structure of the model (cf. [14], p.83) with the delimiters at the beginning and end of each sentence.
- The back-off navigator reproduces the back-off strategy at a very general

level, again without distinguishing specific N-gram histories or calculating probabilities.

- Finally, the robust overlapping concatenation mechanism provides a correct handling of N-gram histories of various length by filtering out illegal ones.

We gave hints for efficient implementations of the five auxiliary WRTs which depend only on Σ and N. All steps are already implemented in the framework of [12].[12]

As previously pointed out, our work is primarily a theoretical one. The huge intermediate automata may prevent its practical application for corpora of the sizes currently used in NLP. Future work will therefore have to concentrate on mechanisms which allow the creation of individual language models for small parts of the underlying corpus and their subsequent combination. In addition, we currently investigate parallel versions of the automata algorithms which exploit multi-processor technology now available.

Another task is the reformulation of state-of-the-art discounting and smoothing methods and the clarification of the relationship between back-off and the other important strategy – interpolation – on a language-theoretic level.

Acknowledgments

We would like to thank Heiko Vogler and our two anonymous reviewers for their critical and very constructive remarks. We are also grateful to Sina Zarrieß, Sebastian Maar and Johannes Bubenzer for very valuable discussions.

References

[1] Aho, Alfred V. and Corasick, Margaret J. Efficient String Matching: An Aid to Bibiographic Search. *Communications of the Asscociation for Computing Machinery*, 18(6):333–340, 1975.

[2] Allauzen, Cyril, Mohri, Mehryar, and Roark, Brian. Generalized Algorithms for Constructing Statistical Language Models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, volume 41, pages 40–47. The Association for Computational Linguistics, 2003.

[3] Bahl, Lalit R., Jelinek, Frederick, and Mercer, Robert L. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Pami-5(2):179–190, 1983.

[4] Bakewell, Adam and Ghica, Dan R. On-the-Fly Techniques for Game-Based Software Model Checking. In Ramakrishnan, C.R. and Rehof, Jakob, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume

[12] All figures in the article were automatically generated by scripts which were processed by the framework's interpreter *fsm2*.

4963 of *Lecture Notes in Computer Science*, pages 78–92. Springer, Berlin, Heidelberg, 2008.

[5] Bullen, Richard H., Jr and Millen, Jonathan K. Microtext: The Design of a Microprogrammed Finite State Search Machine for Full-Text Retrieval. In *Proceedings of the Fall Joint Computer Conference*, AFIPS Joint Computer Conferences, pages 479–488, New York, NY, 1972. ACM.

[6] Chen, Stanley F. and Goodman, Joshua. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, Cambridge, MA, 1998.

[7] de Bruijn, Nicolaas Govert. A Combinatorical Problem. *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen*, 49:758–764, 1946.

[8] Droste, Manfred and Zhang, Guo-Qiang. On Transformations of Formal Power Series. *Information and Computation*, 184(2):369–383, 2003.

[9] Eisner, Jason. Simpler and More General Minimization for Weighted Finite-State Automata. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, pages 64–71, Morristown, NJ, 2003. Association for Computational Linguistics.

[10] Ésik, Zoltán and Kuich, Werner. Equational Axioms for a Theory of Automata. In Vide, Carlos Martín, Mitrana, Victor, and Păun, Gheorghe, editors, *Formal Languages and Applications*, volume 148 of *Studies in Fuzziness and Soft Computing*, chapter 10, pages 183–196. Springer, Berlin, Heidelberg, 2004.

[11] Geyken, Alexander. The DWDS Corpus: A Reference Corpus for the German Language of the 20th Century. In Fellbaum, Christiane, editor, *Collocations and Idioms: Linguistic, Lexicographic, and Computational Aspects*. Continuum Press, London, 2006.

[12] Hanneforth, Thomas. FSM<2.0> – C++ Library for Manipulating (Weighted) Finite Automata. http://www.ling.uni-potsdam.de/~tom/fsm/, 2004.

[13] Hanneforth, Thomas. A Memory Efficient Epsilon-Removal Algorithm for Weighted Acyclic Finite-State Automata. In Piskorski, Jakub, Watson, Bruce, and Yli-Jyrä, Anssi, editors, *Finite-State Methods and Natural Language Processing - Post-proceedings of the 7th International Workshop FSMNLP 2008*, Frontiers in Artificial Intelligence and Applications, 191, pages 72 – 81, Amsterdam, 2008. IOS Press.

[14] Jelinek, Frederick. *Statistical Methods for Speech Recognition*. Language, Speech and Communication. MIT Press, Cambridge, MA, 1997.

[15] Jelinek, Frederick and Mercer, Robert L. Interpolated Estimation of Markov Source Parameters from Sparse Data. In Gelsema, Edzard S. and Kanal, Laveen N., editors, *Pattern Recognition in Practice*, pages 381–397. North-Holland Publishing Company, Amsterdam, 1980.

[16] Jurafsky, Daniel and Martin, James H. *Speech and Language Processing.* Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, NJ, 2000.

[17] Katz, Slava M. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.

[18] Kucera, Henry and Francis, W. Nelson. *Computational Analysis of Present-day American English.* Brown University Press, Providence, RI, 1967.

[19] Kuich, Werner and Salomaa, Arto. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science.* Springer, Berlin, Heidelberg, 1986.

[20] Markov, Andrey A. An Example of Statistical Investigation in the Text of 'Eugene Onyegin' Illustrating Coupling of Tests in Chains. *Proceedings of the Academy of Science St. Petersburg*, 7:153–162, 1913.

[21] Mohri, Mehryar. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2):269–311, 1997.

[22] Mohri, Mehryar. Minimization Algorithms for Sequential Transducers. *Theoretical Computer Science*, 234:177–201, 2000.

[23] Mohri, Mehryar. Generic Epsilon-Removal and Input Epsilon-Normalization Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*, 13(1):129–143, 2002.

[24] Mohri, Mehryar. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[25] Mohri, Mehryar and Riley, Michael D. A Weight Pushing Algorithm for Large Vocabulary Speech Recognition. In *European Conf. on Speech Communication and Technology, Aalborg, Denmark, Sep. 2001*, pages 1603–1606, 2001.

[26] Pereira, Fernando C.N. and Riley, Michael D. Speech Recognition by Composition of Weighted Finite Automata. In Roche, Emmanuel and Schabes, Yves, editors, *Finite-State Language Processing*, volume 12 of *Language, Speech, and Communication*, chapter 15, pages 433–453. The MIT Press, Cambridge, MA, 1997.

[27] Revuz, Dominique. Minimisation of Acyclic Deterministic Automata in Linear Time. *Theoretical Computer Science*, 92(1):181 – 189, 1992.

[28] Riley, Michael D., Pereira, Fernando C., and Mohri, Mehryar. Transducer Composition for Context-Dependent Network Expansion. In Kokkinakis, George, Fakotakis, Nikos, and Dermatas, Evangelos, editors, *EUROSPEECH '97 - 5th European Conference on Speech Communication and Technology*, pages 1427–1430. ISCA, 1997.

[29] Shannon, Claude Elwood. Prediction and Entropy of Printed English. *Bell Labs Technical Journal*, 30:50–64, 1951.

[30] Witten, Ian H. and Bell, Timothy C. The Zero Frequency Problem: Estimating the Probability of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

A Proofs

Lemma 1 (Correctness of conditional N-gram probabilisation). *Definition 19 computes the conditional probability of each N-gram as a special case of Equation (14) (with $i = N$):*

$$\Pr(w_N|w_1^{N-1}) = \frac{\mathrm{C}(w_1^N)}{\sum\limits_{a\in\Sigma} \mathrm{C}(w_1^{N-1}\cdot a)} \,. \tag{21}$$

Proof.

$$\begin{aligned}
&\mathcal{E}_N^1[\mathcal{C}_N](w_1^N) \\
&= \bigoplus_{x\in\Sigma^\bullet} \left(\mathcal{C}_N(x) \otimes \mathcal{E}_N^1(x, w_1^N)\right) && \text{def. of application} \\
&= \bigoplus_{x\in\Sigma^\bullet} \left(\mathcal{C}_N(x) \otimes (ID(\Sigma^{N-1})\cdot(\Sigma\times\Sigma))(x, w_1^N)\right) && \text{def. of } \mathcal{E}_N^1 \\
&= \bigoplus_{a\in\Sigma} \left(\mathcal{C}_N(w_1^{N-1}\cdot a) \otimes \Sigma^{N-1}(w_1^{N-1}) \otimes (\Sigma\times\Sigma)(a, w_N)\right) && \text{def. ID, } \cdot \\
&= \bigoplus_{a\in\Sigma} \big(\mathcal{C}_N(w_1^{N-1}\cdot a) \\
&\quad \otimes \{(w_1^{N-1})\}(w_1^{N-1}) \otimes \Sigma(a) \otimes \Sigma(w_N)\big) && \text{def. of } \cup \text{ and } \times \\
&= \bigoplus_{a\in\Sigma} \big(\mathcal{C}_N(w_1^{N-1}\cdot a) \\
&\quad \otimes \{(w_1^{N-1})\}(w_1^{N-1}) \otimes \{a\}(a) \otimes \{w_N\}(w_N)\big) && \text{def. of } \cup \\
&= \bigoplus_{a\in\Sigma} \big(\mathcal{C}_N(w_1^{N-1}\cdot a) \otimes \bar{1}\big) && \text{def. of singleton} \\
&= \bigoplus_{a\in\Sigma} \mathcal{C}_N(w_1^{N-1}\cdot a) && \text{neutral element}
\end{aligned}$$

Since both operands of the intersection in Definition 19 have the same language projection, $\otimes$-negation replaces each weight of an N-gram by its multiplicative

inverse, and intersection $\otimes$-multiplies weights, Definition 19 mimics Equation (15). □

Lemma 2 (Correspondence of T and $\mathcal{T}_N$). *Given a WRL* $\mathcal{L} : \Sigma^N \to \mathcal{R}$, $\forall w_1^N \in \Sigma^N : \mathcal{T}_N(\mathcal{L})(w_1^N) = \mathrm{T}(w_1^N)$

Proof.

$$
\begin{aligned}
&\mathcal{T}_N(\mathcal{L})(w_1^N) \\
&= \mathcal{E}_N^1[\pi^L(\mathcal{L})](w_1^N) && \text{def. 26} \\
&= \bigoplus_{x \in \Sigma^*} \left(\pi^L(\mathcal{L})(x) \otimes \mathcal{E}_N^1(x, w_1^N)\right) && \text{def. of application} \\
&= \bigoplus_{x \in \Sigma^*} \left(\pi^L(\mathcal{L})(x) \otimes (ID(\Sigma^{N-1}) \cdot (\Sigma \times \Sigma))(x, w_1^N)\right) && \text{def. of } \mathcal{E}_N^1 \\
&= \bigoplus_{a \in \Sigma} \left(\pi^L(\mathcal{L})(w_1^{N-1} \cdot a) \otimes \Sigma^{N-1}(w_1^{N-1}) \otimes (\Sigma \times \Sigma)(a, w_N)\right) && \text{def. ID, } \cdot \\
&= \bigoplus_{a \in \Sigma} \Big(\pi^L(\mathcal{L})(w_1^{N-1} \cdot a) \\
&\quad \otimes \{(w_1^{N-1})\}(w_1^{N-1}) \otimes \Sigma(a) \otimes \Sigma(w_N)\Big) && \text{def. of } \cup \text{ and } \times \\
&= \bigoplus_{a \in \Sigma} \Big(\pi^L(\mathcal{L})(w_1^{N-1} \cdot a) \\
&\quad \otimes \{(w_1^{N-1})\}(w_1^{N-1}) \otimes \{a\}(a) \otimes \{w_N\}(w_N)\Big) && \text{def. of } \cup \\
&= \bigoplus_{a \in \Sigma} \left(\pi^L(\mathcal{L})(w_1^{N-1} \cdot a)\right) && \text{def. singleton, } \bar{1} \\
&= \bigoplus_{a \in \Sigma} \begin{cases} \bar{1} & \text{if } \mathcal{L}(w_1^{N-1} \cdot a) \neq \bar{0} \\ \bar{0} & \text{otherwise .} \end{cases} && \text{def. } \pi^L
\end{aligned}
$$

□

Lemma 4 (Reconstruction of Witten-Bell Discounting). *Given a WRL* $\mathcal{C}_N : \Sigma^N \to \mathcal{R}$, $w_1^N \mapsto \mathrm{C}(w_1^N)$, $\mathcal{W}_N^D(\mathcal{C}_N)(w_1^N)$ *maps an N-gram to its Witten-Bell discounted frequency* $\tilde{\mathrm{C}}(w_1^N)$.

Proof.

$$
\begin{aligned}
&\mathcal{W}_N^D(\mathcal{C}_N)(w_1^N) \\
&= \Big(\mathcal{C}_N \cap \big(\mathcal{N}_N(\mathcal{C}_N) \cap \big(\mathcal{N}_N(\mathcal{C}_N) \cup \mathcal{T}(\mathcal{C}_N)\big)^{-\bar{1}}\big)\Big)(w_1^N) && \text{def. 28} \\
&= \mathcal{C}_N(w_1^N) \otimes \big(\mathcal{N}_N(\mathcal{C}_N)(w_1^N) \otimes \big(\mathcal{N}_N(\mathcal{C}_N)(w_1^N) \oplus \mathcal{T}(\mathcal{C}_N)(w_1^N)\big)^{-\bar{1}}\big) && \text{def. of } \cup \text{ and } \cap \\
&= \mathrm{C}(w_1^N) \otimes \big(\mathrm{N}(w_1^N) \otimes \big(\mathrm{N}(w_1^N) \oplus \mathrm{T}(w_1^N)\big)^{-\bar{1}}\big) && \text{def. of C, N, T}
\end{aligned}
$$

Since $a^{-\bar{1}}$ is $\frac{1}{a}$ in the probability semiring, the last line is equal to Equation (5). The proof for $\mathcal{W}_N^R$ is constructed in the same manner. □

Lemma 5 (Witten-Bell Decomposition). *Given a WRL* $\mathcal{L} : \Sigma^N \to \mathcal{R}$, $\mathcal{W}_N^D(\mathcal{L}) \cup \mathcal{W}_N^R(\mathcal{L}) = \mathcal{L}$.

Proof.

$$
\begin{aligned}
&\mathcal{W}_N^D(\mathcal{L}) \cup \mathcal{W}_N^R(\mathcal{L}) \\
=&\big(\mathcal{L} \cap (\mathcal{N}_N(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\overline{1}})\big) \cup \\
&\big(\mathcal{L} \cap (\mathcal{T}_N(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\overline{1}})\big) && \text{def. of } \mathcal{W}_N^D,\ \mathcal{W}_N^R \\
=&\big((\mathcal{L} \cap \mathcal{N}_N(\mathcal{L})) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\overline{1}}\big) \cup \\
&\big((\mathcal{L} \cap \mathcal{T}_N(\mathcal{L})) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\overline{1}}\big) && \text{assoc. of } \cap \\
=&((\mathcal{L} \cap \mathcal{N}_N(\mathcal{L})) \cup (\mathcal{L} \cap \mathcal{T}_N(\mathcal{L}))) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\overline{1}} && \cap \succ \cup \\
=&\mathcal{L} \cap \big((\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L})) \cap (\mathcal{N}_N(\mathcal{L}) \cup \mathcal{T}_N(\mathcal{L}))^{-\overline{1}}\big) && \cap \succ \cup \\
=&\bigoplus_{x \in \mathcal{L}} \Big(\mathcal{L}(x) \otimes \\
&\big((\mathcal{N}_N(\mathcal{L})(x) \oplus \mathcal{T}_N(\mathcal{L})(x)) \otimes (\mathcal{N}_N(\mathcal{L})(x) \oplus \mathcal{T}_N(\mathcal{L})(x))^{-\overline{1}}\big)\Big) && \text{def. of } \mathcal{L},\ \cup,\ \cap \\
=&\bigoplus_{x \in \mathcal{L}} \big(\mathcal{L}(x) \otimes \overline{1}\big) && \text{def. of } {}^{-\overline{1}} \\
=&\mathcal{L} && \text{def. of } \mathcal{L},\ \overline{1}
\end{aligned}
$$

□

Lemma 7 (Union of $\mathcal{P}_N^{c,D}$ and $\mathcal{P}_N^{c,R}$). *Let* $\mathcal{L}$ *denote a WRL* $\Sigma^N \to \mathcal{R}$*:*

$$\mathcal{P}_N^{c,D}(\mathcal{L}) \cup \mathcal{P}_N^{c,R}(\mathcal{L}) = \mathcal{P}_N^{c}(\mathcal{L}) \ .$$

Proof.

$$
\begin{aligned}
&\mathcal{P}_N^{c,D}(\mathcal{L}) \cup \mathcal{P}_N^{c,R}(\mathcal{L}) \\
&\qquad = (\mathcal{W}_N^D(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}))^{-\overline{1}}) \cup (\mathcal{W}_N^R(\mathcal{L}) \cap (\mathcal{N}_N(\mathcal{L}))^{-\overline{1}}) && \text{by def. 29} \\
&\qquad = (\mathcal{W}_N^D(\mathcal{L}) \cup \mathcal{W}_N^R(\mathcal{L})) \cap (\mathcal{N}_N(\mathcal{L}))^{-\overline{1}} && \text{by } \cap \succ \cup \\
&\qquad = \mathcal{L} \cap (\mathcal{N}_N(\mathcal{L}))^{-\overline{1}} && \text{by lem. 5} \\
&\qquad = \mathcal{L} \cap (\mathcal{E}_N^1[\mathcal{L}])^{-\overline{1}} && \text{by def. 27} \\
&\qquad = \mathcal{P}_N^{c}(\mathcal{L}) && \text{by def. 19}
\end{aligned}
$$

□

Theorem 2 (Robust language model $\mathcal{M}_N^\phi$). *Given* $\mathcal{M}_N^\phi(\mathcal{L})(w)$ *as defined in Definition 34,* $\mathcal{M}_N^\phi(\mathcal{L})(w)$ *computes the correct probability for a delimited input string* w *after equations (20), (7) and (9).*

Proof. Introductory remarks:

- For reasons of simplicity, we restrict the proof to the case of $N = 3$. Proofs for other values for N are analogous.
- Observe that the length of all strings in $\mathcal{Y}_N(\mathcal{L})^*$, $\mathcal{U}_N^\alpha$, $\mathcal{B}_N$ and $\pi^1(\mathcal{D}_N^\phi)$ is a multiple of N. The back-off navigator $\mathcal{B}_N$ places the most specific constraints on the form of the strings to which $\mathcal{D}_N^\phi$ is applied. Thus, in the trigram case, the shortest strings in $\mathcal{Y}_3(\mathcal{L})^* \cap \mathcal{B}_3$ (besides ε, which is ruled out by $\mathcal{U}_N^\alpha$) are of one of the following forms: Σ^3 or $\Sigma^2\alpha \cdot \alpha\Sigma^2$ or $\Sigma^2\alpha \cdot \alpha\Sigma\alpha \cdot \alpha^2\Sigma \cdot \alpha\Sigma^2$.
- Note that the length of ϕ is 0.
- For the reader's better understanding, we spell out the three different cases of Equation (7) for $N = 3$:

$$\hat{\Pr}(w_i|w_{i-2}^{i-1}) = \begin{cases} \tilde{\Pr}(w_i|w_{i-2}^{i-1}) & \text{if } \mathrm{C}(w_{i-2}^i) > 0 \\ \alpha(w_{i-2}^{i-1}) \cdot \tilde{\Pr}(w_i|w_{i-1}) & \text{if } \mathrm{C}(w_{i-2}^i) = 0 \;\&\; \mathrm{C}(w_{i-1}^i) > 0 \\ \alpha(w_{i-2}^{i-1}) \cdot \alpha(w_{i-1}) \cdot \Pr(w_i) & \text{otherwise} \end{cases} . \tag{22}$$

  Remember that the values of the αs in Equation (22) may be 1 in case the history is not present (cf. Equation (9)).

Since $\mathcal{U}_3^\alpha$ introduces the sentence delimiters, the proof is by induction on the length of the string $w = \langle s\rangle^2 w' \langle /s\rangle^2$.

Induction hypothesis:
Let $w_1^k = \langle s\rangle^2 w' \langle /s\rangle^2$ an input string of length $k \geq 4$ $(= 2(N-1))$:

$$\mathcal{D}_3^\phi[\mathcal{Y}_3(\mathcal{L})^* \cap \mathcal{U}_3^\alpha \cap \mathcal{B}_3](w_1^k) = \prod_{i=3=N}^{k} \hat{\Pr}(w_i|w_{i-2}^{i-1}) . \tag{23}$$

Induction base: $|w| = 0$.
Case 1a: $w = \varepsilon$ (this means that the trigram $\langle s\rangle^2\langle /s\rangle$ is in $\mathcal{Y}_3(\mathcal{L})$)

$$\begin{aligned}
&\mathcal{D}_3^\phi[(\mathcal{Y}_3)^*](\langle s\rangle^2\langle /s\rangle^2) \\
&= \bigoplus_{x\in\Sigma^*} ((\mathcal{Y}_3^*)(x) \otimes \mathcal{D}_3^\phi(x, \langle s\rangle^2\langle /s\rangle^2)) && \text{def. of appl.} \\
&= (\mathcal{Y}_3^*)(\langle s\rangle^2\langle /s\rangle \cdot \langle s\rangle\langle /s\rangle^2) \\
&\quad \otimes (\Sigma^3(\langle s\rangle^2\langle /s\rangle) \\
&\quad \otimes \{(\langle s\rangle\langle /s\rangle^2, \langle /s\rangle)\}(\langle s\rangle\langle /s\rangle^2, \langle /s\rangle)) && \text{def. of } \mathcal{D}_3^\phi \\
&= (\mathcal{Y}_3^*)(\langle s\rangle^2\langle /s\rangle \cdot \langle s\rangle\langle /s\rangle^2) && \text{ID, singleton, } \bar{1} \\
&= (\mathcal{Y}_3)(\langle s\rangle^2\langle /s\rangle) \otimes (\mathcal{Y}_3)(\langle s\rangle\langle /s\rangle^2) && \text{closure} \\
&= (\mathcal{P}_3^{c,D})(\langle s\rangle^2\langle /s\rangle) \otimes (\mathcal{P}_3^{c,D})(\langle s\rangle\langle /s\rangle^2) && \text{def. of } \mathcal{Y}_3
\end{aligned}$$

$$=\tilde{\Pr}(\text{</s>}\,|\,\text{<s>}^2) \otimes \tilde{\Pr}(\text{</s>}\,|\,\text{<s></s>}) \qquad \text{lem. 6}$$
$$=\hat{\Pr}(\text{</s>}\,|\,\text{<s>}^2) \otimes \hat{\Pr}(\text{</s>}\,|\,\text{<s></s>}) \qquad \text{eqs. (20), (22, case 1)}$$

Case 1b: $w = \phi\phi$ (this means that the trigram $\text{<s>}^2\text{</s>}$ is not in $\mathcal{Y}_3(\mathcal{L})$, which in turn entails that the trigram $\alpha\ \text{<s></s>}$ will also not be present in $\mathcal{Y}_3(\mathcal{L})$). $\mathcal{D}_3^{\phi}$ will decompose $\text{<s>}^2\,\phi\phi\,\text{</s>}^2$ into

$$(\text{<s>}^2\,\alpha, \text{<s>}^2\,\phi) \cdot (\alpha\,\text{<s>}\,\alpha, \phi) \cdot (\alpha^2\,\text{</s>}, \text{</s>}) \cdot (\alpha\,\text{</s>}^2, \text{</s>})$$

whose first projection is also in $\mathcal{U}_3^{\alpha} \cap \mathcal{B}_3$.

$$\mathcal{Y}_3^*(\text{<s>}^2\,\alpha \cdot \alpha\,\text{<s>}\,\alpha \cdot \alpha^2\,\text{</s>} \cdot \alpha\,\text{</s>}^2)$$
$$=\mathcal{Y}_3(\text{<s>}^2\,\alpha) \otimes \mathcal{Y}_3(\alpha\,\text{<s>}\,\alpha) \otimes \mathcal{Y}_3(\alpha^2\,\text{</s>}) \otimes \mathcal{Y}_3(\alpha\,\text{</s>}^2) \qquad \text{closure}$$
$$=\mathcal{P}_3^{c,R}(\text{<s>}^2\,\alpha) \otimes \mathcal{P}_2^{c,R}(\text{<s>}\,\alpha) \otimes \mathcal{P}_1^{c}(\text{</s>}) \otimes \mathcal{P}_2^{c,D}(\text{</s>}^2) \qquad \text{def. 30}$$
$$=\big(\alpha(\text{<s>}^2) \otimes \alpha(\text{<s>}) \otimes \Pr(\text{</s>})\big) \otimes \tilde{\Pr}(\text{</s>}|\text{</s>}) \qquad \text{lem. 9, lem. 6}$$
$$=\hat{\Pr}(\text{</s>}|\text{<s>}^2) \otimes \tilde{\Pr}(\text{</s>}|\text{</s>}) \qquad \text{eq. (22, case 3)}$$
$$=\hat{\Pr}(\text{</s>}|\text{<s>}^2) \otimes \big(\overline{1} \otimes \tilde{\Pr}(\text{</s>}|\text{</s>})\big) \qquad \text{neutr. element of } \otimes$$
$$=\hat{\Pr}(\text{</s>}|\text{<s>}^2) \otimes \hat{\Pr}(\text{</s>}|\text{<s></s>}) \qquad \text{eqs. (9), (22, case 2)}$$

Induction step: Assume, the induction hypothesis holds for strings $w_1^i = \text{<s>}^2\ w'\ \text{</s>}^2$ with $4\ (=2(N-1)) \leq i \leq k$. We show that it also holds for $k+1$. For the proof, there are two possible cases concerning the history w_{k-1}^k of w_{k+1}:[13]

1. $w_1^k = w'' w_{k-1}^k$ or $w_1^k = w'' w_{k-1}\ \phi\ w_k$: the history w_{k-1}^k is present. Here we have three subcases, depending on in which distribution we successfully process w_{k+1}:
   a) $w_1^{k+1} = w_1^k w_{k+1}$: trigrams
   b) $w_1^{k+1} = w_1^k \phi w_{k+1}$: bigrams
   c) $w_1^{k+1} = w_1^k \phi\phi w_{k+1}$: unigrams
2. $w_1^k = w'' w_{k-1}\ \phi\phi\ w_k$: the history w_{k-1}^k is not present, since w_k was processed (after reading two occurrences of ϕ) in the unigram distribution. Here we have two subcases to consider, which can only occur after case 1c) above or 2b) below:
   a) $w_1^{k+1} = w_1^{k-1}\ \phi\phi\ w_k w_{k+1}$: bigrams (superdistribution)
   b) $w_1^{k+1} = w_1^{k-1}\ \phi\phi\ w_k\ \phi\ w_{k+1}$: unigrams

In the following, we give proofs for the 5 subcases mentioned above.

[13] *Remark*: With respect to the back-off navigator in Figure 11, this distinction is reflected in the particular state the navigator is after having processed w_k: In the first case, this state is 0, while it is 9 in the second. In the general case of an N-gram navigator, there will be $N-1$ such states, and in turn $N-1$ main cases to consider.

1. a) $\mathcal{D}_3^{\phi}$ maps w_{k+1} to w_{k-1}^{k+1}, which is also in $\mathcal{B}_3$.

$$\begin{aligned}
&\mathcal{Y}_3(w_{k-1}^{k+1}) \\
=&\mathcal{P}_3^{c,D}(w_{k-1}^{k+1}) && \text{def. 30} \\
=&\tilde{\Pr}(w_{k+1}|w_{k-1}^{k}) && \text{lem. 6} \\
=&\hat{\Pr}(w_{k+1}|w_{k-1}^{k}) && \text{eq. (22, case 1)}
\end{aligned}$$

b) $\mathcal{D}_3^{\phi}$ maps ϕw_{k+1} to $w_{k-1}^{k}\alpha \cdot \alpha w_k^{k+1}$, which is also in $\mathcal{B}_3$.

$$\begin{aligned}
&\mathcal{Y}_3^*(w_{k-1}^{k}\alpha \cdot \alpha w_k^{k+1}) \\
=&\mathcal{Y}_3(w_{k-1}^{k}\alpha) \otimes \mathcal{Y}_3(\alpha w_k^{k+1}) && \text{def. of closure} \\
=&\mathcal{P}_3^{c,R}(w_{k-1}^{k}\alpha) \otimes \mathcal{P}_2^{c,D}(w_k^{k+1}) && \text{def. 30} \\
=&\alpha(w_{k-1}^{k}) \otimes \tilde{\Pr}(w_{k+1}|w_k) && \text{lem. 9, lem. 6} \\
=&\hat{\Pr}(w_{k+1}|w_{k-1}^{k}) && \text{eq. (22, case 2)}
\end{aligned}$$

c) $\mathcal{D}_3^{\phi}$ maps $\phi\phi w_{k+1}$ to $w_{k-1}^{k}\alpha \cdot \alpha w_k \alpha \cdot \alpha^2 w_{k+1}$. This string is not in $\mathcal{B}_3$, but is a prefix of one of its strings, namely $w_{k-1}^{k}\alpha \cdot \alpha w_k \alpha \cdot \alpha^2 w_{k+1} \cdot \alpha w_k^{k+1}$. The "missing" suffix αw_k^{k+1} will be covered in case 2a) or 2b), which are the only possible cases following 1c).

$$\begin{aligned}
&\mathcal{Y}_3^*(w_{k-1}^{k}\alpha \cdot \alpha w_k \alpha \cdot \alpha^2 w_{k+1}) \\
=&\mathcal{Y}_3(w_{k-1}^{k}\alpha) \otimes \mathcal{Y}_3(\alpha w_k \alpha) \otimes \mathcal{Y}_3(\alpha^2 w_{k+1}) && \text{def. of closure} \\
=&\mathcal{P}_3^{c,R}(w_{k-1}^{k}\alpha) \otimes \mathcal{P}_2^{c,R}(w_k \alpha) \otimes \mathcal{P}_1^{c}(w_{k+1}) && \text{def. 30} \\
=&\alpha(w_{k-1}^{k}) \otimes \alpha(w_k) \otimes \Pr(w_{k+1}) && \text{lem. 9, lem. 6} \\
=&\hat{\Pr}(w_{k+1}|w_{k-1}^{k}) && \text{eq. (22, case 3)}
\end{aligned}$$

2. The following subcases cover the "missing" suffix of case 1c) above.

a) $\mathcal{D}_3^{\phi}$ maps w_{k+1} to αw_k^{k+1}.

$$\begin{aligned}
&\mathcal{Y}_3^*(\alpha w_k^{k+1}) \\
=&\mathcal{Y}_3(\alpha w_k^{k+1}) && \text{def. of closure} \\
=&\mathcal{P}_2^{c,D}(w_k^{k+1}) && \text{def. 30} \\
=&\tilde{\Pr}(w_{k+1}|w_k) && \text{lem. 6} \\
=&\overline{1} \otimes \tilde{\Pr}(w_{k+1}|w_k) && \text{eq. (9, case 2)} \\
=&\hat{\Pr}(w_{k+1}|w_{k-1}^{k}) && \text{eq. (22, case 2)}
\end{aligned}$$

b) $\mathcal{D}_3^{\phi}$ maps ϕw_{k+1} to $\alpha w_k \alpha \cdot \alpha^2 w_{k+1}$.

$$\mathcal{Y}_3^*(\alpha w_k \alpha \cdot \alpha^2 w_{k+1})$$

$$
\begin{aligned}
&= \mathcal{Y}_3(\alpha w_k \dot{\alpha}) \otimes \mathcal{Y}_3(\alpha^2 w_{k+1}) && \text{def. of closure} \\
&= \mathcal{P}_2^{c,R}(w_k \alpha) \otimes \mathcal{P}_1^{c}(w_{k+1}) && \text{def. 30} \\
&= \alpha(w_k) \otimes \Pr(w_{k+1}) && \text{lem. 9, lem. 6} \\
&= \overline{1} \otimes \alpha(w_k) \otimes \Pr(w_{k+1}) && \text{eq. (9, case 2)} \\
&= \hat{\Pr}(w_{k+1} | w_{k-1}^k) && \text{eq. (22, case 3)}
\end{aligned}
$$

Combining this with the induction hypothesis, we get

$$
\mathcal{D}_3^{\phi}[\mathcal{Y}_3(\mathcal{L})^* \cap \mathcal{U}_3^{\alpha} \cap \mathcal{B}_3](w_1^{k+1}) = \prod_{i=N}^{k+1} \hat{\Pr}(w_i | w_{i-2}^{i-1}) \; . \tag{24}
$$

Note that the $N-1$ sentence delimiters </s> ensure that the N-grams α </s>$^{N-1}$ or a </s>$^{N-1}$ for some $a \in \Sigma$ are always present in $\mathcal{Y}_N$ such that the last step of the computation of the decomposed back-off probability of an delimited input sentence will always be case 1a) or case 2a). □

B Constructions

Definition 35 (Construction of $\mathfrak{F}_N$). *The weighted finite state transducer $\mathfrak{F}_N$ wrt a semiring $\mathcal{R}$ is an 8-tuple $\langle Q, \Sigma, \Sigma \cup \{\varepsilon\}, 0, F, E_i \cup E_m \cup E_f, \overline{1}, \rho \rangle$ where*

$$
\begin{aligned}
Q &= \bigcup_{i=0}^{N} \{i\} \\
F &= \{N\} \\
E_i &= \bigcup_{a \in \Sigma} \{(0, 0, a, \varepsilon, \overline{1})\} \cup \bigcup_{a \in \Sigma} \{(0, 1, a, a, \overline{1})\} \\
E_m &= \bigcup_{i=1}^{N-1} \bigcup_{a \in \Sigma} \{(i, i+1, a, a, \overline{1})\} \\
E_f &= \bigcup_{a \in \Sigma} \{(N, N, a, \varepsilon, \overline{1})\} \\
\forall q \in F, \rho(q) &= \overline{1}
\end{aligned}
$$

Definition 36 (Construction of $\mathfrak{E}_N^k$). *The weighted finite state transducer $\mathfrak{E}_N^k$ wrt*

a semiring $\mathcal{R}$ is an 8-tuple $\langle Q, \Sigma, \Sigma, 0, F, E_m \cup E_k, \overline{1}, \rho\rangle$ where

$$Q = \bigcup_{i=0}^{N} \{i\}$$
$$F = \{N\}$$
$$E_m = \bigcup_{i=0}^{N-k-1} \bigcup_{a\in\Sigma} \{(i, i+1, a, a, \overline{1})\}$$
$$E_k = \bigcup_{i=N-k}^{N-1} \bigcup_{a\in\Sigma} \bigcup_{b\in\Sigma} \{(i, i+1, a, b, \overline{1})\}$$
$$\forall q \in F, \rho(q) = \overline{1}$$

Definition 37 (Construction of $\mathfrak{D}_N$). *The weighted finite state transducer $\mathfrak{D}_N$ wrt a semiring $\mathcal{R}$ is an 8-tuple $\langle Q, \Sigma, \Sigma, 0, F, E_0 \cup E_t \cup E_o \cup E_l, \overline{1}, \rho\rangle$ where (using q_y from Definition 25):*[14]

$$Q = \bigcup_{i=0}^{q_y+|\Sigma|^{N-1}*(N-1)-1} \{i\}$$
$$F = \bigcup_{i=q_y}^{q_y+|\Sigma|^{N-1}-1} \{i\}$$
$$E_0 = \bigcup_{a\in\Sigma} \{(0, 1, a)\}$$
$$E_t = \bigcup_{i=1}^{q_y-1} \bigcup_{a\in\Sigma} \left\{\left(i, \left(i * |\Sigma| + \mathtt{idx}(a)\right) - \left(|\Sigma| - 2\right), a, a, \overline{1}\right)\right\}$$
$$E_o = \bigcup_{i=q_y}^{q_y+|\Sigma|^{N-1}*(N-2)-1} \left\{(i, i+|\Sigma|^{N-1}, a, \varepsilon, \overline{1}) \middle| \mathtt{idx}(a) = \left\lfloor \frac{i-q_y}{|\Sigma|^{N-2-\left\lfloor \frac{i-q_y}{|\Sigma|^{N-1}} \right\rfloor}} \right\rfloor \bmod |\Sigma| \right\}$$
$$E_l = \bigcup_{i=q_y+|\Sigma|^{N-1}*(N-2)}^{q_y+|\Sigma|^{N-1}*(N-1)-1} \{(i, \mathtt{idx}(a)+2, a, \varepsilon, \overline{1}) | \mathtt{idx}(a) = (i-q_y) \bmod |\Sigma|\}$$
$$\rho(q) = \overline{1}, \forall q \in F.$$

Received 15th August 2008

[14] $\lfloor x \rfloor$ denotes the *floor* value of a number. E.g. $\lfloor 2.34 \rfloor = 2$.

Acta Cybernetica 19 (2009) 357–392.

Max/Plus Tree Automata for Termination of Term Rewriting

Adam Koprowski* and Johannes Waldmann†

Abstract

We use weighted tree automata as certificates for termination of term rewriting systems. The weights are taken from the arctic semiring: natural numbers extended with $-\infty$, with the operations "max" and "plus". In order to find and validate these certificates automatically, we restrict their transition functions to be representable by matrix operations in the semiring. The resulting class of weighted tree automata is called *path-separated*.

This extends the matrix method for term rewriting and the arctic matrix method for string rewriting. In combination with the dependency pair method, this allows for some conceptually simple termination proofs in cases where only much more involved proofs were known before. We further generalize to arctic numbers "below zero": integers extended with $-\infty$. This allows to treat some termination problems with symbols that require a predecessor semantics.

Correctness of this approach has been formally verified in the Coq proof assistant and the formalization has been contributed to the CoLoR library of certified termination techniques. This allows formal verification of termination proofs using the arctic matrix method in combination with the dependency pair transformation. This contribution brought a substantial performance gain in the certified category of the 2008 edition of the termination competition.

The method has been implemented by leading termination provers. We report on experiments with its implementation in one such tool, Matchbox, developed by the second author.

We also show that our method can simulate a previous method of quasi-periodic interpretations, if restricted to interpretations of slope one on unary signatures.

Keywords: term rewriting, termination, weighted tree automaton, max/plus algebra, arctic semiring, monotone algebra, matrix interpretation, formal verification

*Institute for Computing and Information Science, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands *and* MLstate, rue Berlier 15, 75013 Paris, France, E-mail: Adam.Koprowski@cs.ru.nl

†Hochschule für Technik, Wirtschaft und Kultur Leipzig, Fb IMN, PF 30 11 66, D-04251 Leipzig, Germany, E-mail: waldmann@imn.htwk-leipzig.de

1 Introduction

One method of proving termination is interpretation into a well-founded algebra. Polynomial interpretations (over the naturals) are a well-known example of this approach. Another example is the recent development of the matrix method [22, 13] that uses linear interpretations over vectors of naturals, or equivalently, $\mathbb{N}$-weighted automata. In [38, 37] one of the authors extended this method (for string rewriting) to arctic automata, *i.e.*, on the max/plus semiring on $\{-\infty\} \cup \mathbb{N}$. Its implementation in the termination prover Matchbox [36] contributed to this prover winning the string rewriting division of the 2007 termination competition [31, 1].

The first contribution of the present work is a *generalization of arctic termination to term rewriting.* We use interpretations given by functions of the form $(\vec{x}_1, \ldots, \vec{x}_n) \mapsto M_1 \cdot \vec{x}_1 + \ldots + M_n \cdot \vec{x}_n + \vec{c}$. Here, $\vec{x}_i$ are (column) vector variables, $\vec{c}$ is a vector and $M_1, \ldots, M_n$ are square matrices, where all entries are arctic numbers, and operations are understood in the arctic semiring.

Functions of this shape compute the transition function of a weighted tree automaton [10, 9]. The vectors correspond to assignments from states to weights.

Since the max operation is not strictly monotone in single arguments, we obtain monotone interpretations only for the case when all function symbols are at most unary, *i.e.*, string rewriting. For symbols of higher arity, arctic interpretations are weakly monotone. These cannot prove termination, but only top termination, where rewriting steps are only applied at the root of terms. This is a restriction but it fits with the framework of the dependency pair method [4] that transforms a termination problem to a top termination problem.

The second contribution is a *generalization from arctic naturals to arctic integers*, *i.e.*, $\{-\infty\} \cup \mathbb{Z}$. Arctic integers allow for example to interpret function symbols by the predecessor function and this matches the "intrinsic" semantics of some termination problems. There is previous work on polynomial interpretations with negative coefficients [19, 20], where the interpretation for predecessor is also expressible using ad-hoc max operations. Using arctic integers, we obtain verified termination proofs for 10 of the 24 rewrite systems Beerendonk/* from the Termination Problem Database [2] (TPDB), simulating imperative computations. Previously, they could only be handled by the method of Bounded Increase [17] and polynomial interpretations with rational coefficients [30].

The third contribution is that we can express *quasi-periodic interpretations* [39] of slope one, another powerful method for proving termination of rewriting, as an instance of arctic interpretations for unary signatures.

The next contribution is the fact that the correctness of this method for proving top termination has been *formally verified with the proof assistant Coq* [34]. This extends previous work [27] and is now part of the CoLoR project [7] that gathers formalizations of termination techniques and employs them to certify proofs found by tools for automatic termination proving. This contribution was crucial in enabling CoLoR to win against the competing certification back-end, A3PAT [8], in the termination competition of 2008 [1].

A method to search for arctic interpretations is implemented for the termination prover Matchbox. It works by transformation to a boolean satisfiability problem and application of a state-of-the-art SAT solver (in this case, Minisat [11]). For a number of problems Matchbox produced certified termination proofs, where only un-certified proofs were available before. Recently the arctic interpretations method was also implemented in AProVE [16] and T_TT_2 [28].

The paper is organized as follows. We present notation and basic facts on rewriting in Section 2 and give an introduction to proving termination of rewriting using the monotone algebra framework in Section 3. Then we give preliminaries on the arctic semiring in Section 4, and we relate the monotone algebra approach to the concept of weighted tree automata in Section 5. We present arctic interpretations for termination in Section 6, for top termination in Section 7 and the generalization to arctic integers in Section 8. In Section 9 we show that quasi-periodic interpretations of slope one for proving termination of string rewriting [39] are a special case of arctic matrix interpretations. We report on the formal verification in Section 10 and on performance of our implementation in Section 11. We present some discussion of the method, its limitations and related work in Section 12 and we conclude in Section 13.

Preliminary versions of the results from this paper have been presented at the Workshop on Termination [37], at the Workshop on Weighted Automata [26], and at the Conference on Rewriting Techniques and Applications [25]. We thank the anonymous referees for their comments.

2 Term Rewriting

In this section we shortly introduce the basic notions on term rewriting. For more details we refer to [5].

Let Σ be a signature, that is, a set of operation symbols each having a fixed arity in $\mathbb{N}$. For a set of variable symbols $\mathcal{V}$, disjoint from Σ, let $\mathcal{T}(\Sigma, \mathcal{V})$ be the set of terms over Σ and $\mathcal{V}$, that is, the smallest set satisfying

- $x \in \mathcal{T}(\Sigma, \mathcal{V})$ for all $x \in \mathcal{V}$, and
- if the arity of $f \in \Sigma$ is n and $t_i \in \mathcal{T}(\Sigma, \mathcal{V})$ for $i = 1, \ldots, n$, then $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$.

Terms are identified with finitely branching labeled trees. We denote a root of a term t by $\mathrm{root}(t)$ and $\mathrm{root}(f(t_1, \ldots, t_n)) = f$. By $\trianglelefteq$ we denote the sub-term relation on terms and we have $t \trianglelefteq u$ if t is a sub-tree of u.

A *term rewriting system* (TRS) $\mathcal{R}$ over $\Sigma, \mathcal{V}$ is a set of pairs $(\ell, r) \in \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$, for which $\ell \notin \mathcal{V}$ and all variables in r occur in ℓ. Pairs (ℓ, r) are called *rewrite rules* and are usually written as $\ell \to r$.

A TRS with all functions having arity one is called a *string rewriting system* (SRS). For SRSs it is customary to write terms as strings, so $a_1(a_2(\ldots(a_n(x))\ldots))$ becomes $a_1a_2\cdots a_n$ and x becomes ϵ.

For a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ and a term t the application of σ to t, denoted by $t\sigma$, is a term defined inductively as:

- $x\sigma = \sigma(x)$ for all $x \in \mathcal{V}$, and
- $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$.

For a TRS $\mathcal{R}$ the *top rewrite relation* $\xrightarrow{\text{top}}_{\mathcal{R}}$ on $\mathcal{T}(\Sigma, \mathcal{V})$ is defined by $t \xrightarrow{\text{top}}_{\mathcal{R}} u$ if and only if there is a rewrite rule $\ell \to r \in \mathcal{R}$ and a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ such that $t = \ell\sigma$ and $u = r\sigma$. The *rewrite relation* $\to_{\mathcal{R}}$ is defined to be the smallest relation satisfying

- if $t \xrightarrow{\text{top}}_{\mathcal{R}} u$ then $t \to_{\mathcal{R}} u$, and
- if $t_i \to_{\mathcal{R}} u_i$ and $t_j = u_j$ for $j \neq i$, then $f(t_1, \ldots, t_n) \to_{\mathcal{R}} f(u_1, \ldots, u_n)$ for every $f \in \Sigma$ of arity n.

A relation $\to$ is terminating if it does not admit infinite descending chains $t_0 \to t_1 \to \ldots$, denoted as $\text{SN}(\to)$. For relations $\to_1, \to_2$, we define $\to_1 \,/\, \to_2$ by $(\to_1) \cdot (\to_2)^*$. If $\text{SN}(\to_1 \,/\, \to_2)$, we say that $\to_1$ is terminating relative to $\to_2$.

When given as arguments to SN we will often identify TRSs with the rewrite relations generated by them and hence abbreviate $\xrightarrow{\text{top}}_{\mathcal{R}}$ by $\mathcal{R}_{\text{top}}$ and $\to_{\mathcal{R}}$ by $\mathcal{R}$.

Now we will shortly introduce the dependency pair method [4] — a powerful approach for proving termination of rewriting, used by most of the termination provers.

Definition 2.1. [Dependency pairs] Let $\mathcal{R}$ be a TRS over a signature Σ. The set of *defined* symbols is defined as $\mathcal{D}_{\mathcal{R}} = \{\text{root}(l) \mid l \to r \in \mathcal{R}\}$. We extend a signature Σ to the signature $\Sigma^\sharp$ by adding symbols $f^\sharp$ for every symbol $f \in \mathcal{D}_{\mathcal{R}}$. If $t \in \mathcal{T}(\Sigma, \mathcal{V})$ with $\text{root}(t) \in \mathcal{D}_{\mathcal{R}}$ then $t^\sharp$ denotes the term that is obtained from t by replacing its root symbol with $\text{root}(t)^\sharp$.

If $l \to r \in \mathcal{R}$ and $t \trianglelefteq r$ with $\text{root}(t) \in \mathcal{D}_{\mathcal{R}}$ then the rule $l^\sharp \to t^\sharp$ is a *dependency pair* of $\mathcal{R}$. The set of all dependency pairs of $\mathcal{R}$ is denoted by $\text{DP}(\mathcal{R})$. ◇

The main theorem underlying the dependency pair method is the following.

Theorem 2.2 ([4]). Let $\mathcal{R}$ be a TRS. $\text{SN}(\mathcal{R})$ iff $\text{SN}(\text{DP}(\mathcal{R})_{\text{top}}/\mathcal{R})$. □

In this paper we will consider problems of termination of rewrite relations generated by some term rewriting systems. Three types of problems will be of interest:

- Full termination: *given a TRS $\mathcal{R}$, is it terminating, i.e., does* $\text{SN}(\mathcal{R})$ *hold?*
- Relative termination: *given two TRSs $\mathcal{R}, \mathcal{S}$, does $\mathcal{R}$ terminate relative to $\mathcal{S}$, i.e., does* $\text{SN}(\mathcal{R}/\mathcal{S})$ *hold?*
- Relative top termination: *given two TRSs $\mathcal{R}, \mathcal{S}$ does $\mathcal{R}$ terminate relative to $\mathcal{S}$ if we allow only top reductions in $\mathcal{R}$, i.e., does* $\text{SN}(\mathcal{R}_{\text{top}}/\mathcal{S})$ *hold?*

Note that termination is a special case of relative termination as $\mathrm{SN}(\mathcal{R}) \iff \mathrm{SN}(\mathcal{R}/\emptyset)$, hence we will present results for relative termination only as they are immediately applicable for the full termination case. Relative top termination is of special interest due to its close relation with the dependency pair method, established in Theorem 2.2.

We will illustrate some term rewriting notions on an example.

Example 2.3. Consider the following three rules TRS $\mathcal{R}$, AG01/#3.41 from the TPDB [2], over the signature $\Sigma = \{0, \mathrm{p}, \mathrm{s}, \mathrm{fac}, \mathrm{times}\}$:

$$\begin{aligned} \mathrm{p}(\mathrm{s}(x)) &\to x \\ \mathrm{fac}(0) &\to \mathrm{s}(0) \\ \mathrm{fac}(\mathrm{s}(x)) &\to \mathrm{times}(\mathrm{s}(x), \mathrm{fac}(\mathrm{p}(\mathrm{s}(x)))) \end{aligned}$$

This TRS represents computation of the factorial function (without the rules for addition and multiplication) with natural numbers represented with zero (0), successor (s) and predecessor (p) and the factorial function (fac) expressed using multiplication (times).

We have the following reduction sequence (with the redex at every step underlined):

$$\begin{aligned} \underline{\mathrm{fac}(\mathrm{s}(\mathrm{s}(0)))} &\overset{\mathrm{top}}{\to}_{\mathcal{R}} \mathrm{times}(\mathrm{s}(\mathrm{s}(0)), \mathrm{fac}(\underline{\mathrm{p}(\mathrm{s}(\mathrm{s}(0)))})) \to_{\mathcal{R}} \\ \mathrm{times}(\mathrm{s}(\mathrm{s}(0)), \underline{\mathrm{fac}(\mathrm{s}(0))}) &\to_{\mathcal{R}} \mathrm{times}(\mathrm{s}(\mathrm{s}(0)), \mathrm{times}(\mathrm{s}(0), \mathrm{fac}(\underline{\mathrm{p}(\mathrm{s}(0))}))) \to_{\mathcal{R}} \\ \mathrm{times}(\mathrm{s}(\mathrm{s}(0)), \mathrm{times}(\mathrm{s}(0), \underline{\mathrm{fac}(0)})) &\to_{\mathcal{R}} \mathrm{times}(\mathrm{s}(\mathrm{s}(0)), \mathrm{times}(\mathrm{s}(0), \mathrm{s}(0))) \end{aligned}$$

calculating that factorial of two equals $2 \times (1 \times 1)$. We also have two defined symbols, $\mathcal{D}_{\mathcal{R}} = \{\mathrm{p}, \mathrm{fac}\}$, extended signature $\Sigma^\sharp = \Sigma \cup \{\mathrm{p}^\sharp, \mathrm{fac}^\sharp\}$ and two dependency pairs:

$$\begin{aligned} \mathrm{fac}^\sharp(\mathrm{s}(x)) &\to \mathrm{fac}^\sharp(\mathrm{p}(\mathrm{s}(x))) \\ \mathrm{fac}^\sharp(\mathrm{s}(x)) &\to \mathrm{p}^\sharp(\mathrm{s}(x)) \end{aligned}$$

We will prove termination of this example in the following section. ◁

3 Monotone Algebras

We will now introduce the definitions and results of monotone algebras, following the presentation of [13].

Definition 3.1. [Monotonicity] Let A be a non-empty set. An operation $[f] : A \times \cdots \times A \to A$ is *monotone* with respect to a binary relation $\to$ on A if for all $a_1, \ldots, a_i, a_i', \ldots a_n \in A$ with $a_i \to a_i'$ we have

$$[f](a_1, \ldots, a_i, \ldots, a_n) \to [f](a_1, \ldots, a_i', \ldots, a_n) \qquad \diamond$$

Definition 3.2. [Σ-algebra] A Σ-*algebra*, $(A, \{f_A\}_{f \in \Sigma})$ consists of a non-empty set A together with a map $[f_A] : A^n \to A$ for every $f \in \Sigma$, where n is the arity of f. $\diamond$

Definition 3.3. [Weakly monotone Σ-algebra] Let $\mathcal{R}$ be a TRS over a signature Σ. A well-founded *weakly monotone* Σ*-algebra* is a quadruple $\mathcal{A} = (A, \{f_A\}_{f \in \Sigma}, >, \gtrsim)$ such that:

- $(A, \{f_A\}_{f \in \Sigma})$ is a Σ-algebra,
- all algebra operations are weakly monotone, *i.e.*, monotone with respect to $\gtrsim$,
- $>$ is a well-founded relation on A, and
- relations $\gtrsim$ and $>$ are compatible, that is: $> \cdot \gtrsim \subseteq >$ or $\gtrsim \cdot > \subseteq >$.

An *extended monotone* Σ*-algebra* $(A, \{f_A\}_{f \in \Sigma}, >, \gtrsim)$ is a weakly monotone Σ-algebra $(A, \{f_A\}_{f \in \Sigma}, >, \gtrsim)$ in which moreover for every $f \in \Sigma$ the operation $[f]$ is strictly monotone, *i.e.*, monotone with respect to $>$. ◇

Definition 3.4. For a weakly monotone Σ-algebra $\mathcal{A} = (A, \{f_A\}_{f \in \Sigma}, >, \gtrsim)$ we extend the order $\gtrsim$ on A to an order $\gtrsim_\alpha$ on terms, as

$$t \gtrsim_\alpha u \iff \forall_{\alpha : \mathcal{V} \to A} : \ [t]_\alpha \gtrsim [u]_\alpha$$

$>$ is extended to $>_\alpha$ in a similar way. ◇

Now we present a slight variant of the main theorem from [13], for proving relative (top)-termination with monotone algebras:

Theorem 3.5. Let $\mathcal{R}, \mathcal{R}', \mathcal{S}, \mathcal{S}'$ be TRSs over a signature Σ.

(a) Let $(A, [\cdot], >, \gtrsim)$ be an extended monotone algebra such that: $[\ell] \gtrsim_\alpha [r]$ for every rule $\ell \to r \in \mathcal{R} \cup \mathcal{S}$ and $[\ell] >_\alpha [r]$ for every rule $\ell \to r \in \mathcal{R}' \cup \mathcal{S}'$. Then $\mathrm{SN}(\mathcal{R}/\mathcal{S})$ implies $\mathrm{SN}(\mathcal{R} \cup \mathcal{R}'/\mathcal{S} \cup \mathcal{S}')$.

(b) Let $(A, [\cdot], >, \gtrsim)$ be a weakly monotone algebra such that: $[\ell] \gtrsim_\alpha [r]$ for every rule $\ell \to r \in \mathcal{R} \cup \mathcal{S}$ and $[\ell] >_\alpha [r]$ for every rule $\ell \to r \in \mathcal{R}'$. Then $\mathrm{SN}(\mathcal{R}_{\mathrm{top}}/\mathcal{S})$ implies $\mathrm{SN}(\mathcal{R}_{\mathrm{top}} \cup \mathcal{R}'_{\mathrm{top}}/\mathcal{S})$. □

We will illustrate the application of this theorem on a simple example using the matrix interpretation method [13].

Example 3.6. Consider the TRS from Example 2.3. We will show how Theorem 3.5a can be applied to this TRS in order to simplify the related termination problem.

For that we first need to choose a suitable monotone algebra. For the domain A we take vectors over $\mathbb{N}$ of length 2 with the following orders:

$$(u_1, u_2) \gtrsim (v_1, v_2) \iff u_1 \geq v_1 \wedge u_2 \geq v_2$$
$$(u_1, u_2) > (v_1, v_2) \iff u_1 > v_1 \wedge u_2 \geq v_2$$

Compatibility of those orders and well-foundedness of $>$ are immediate. For interpretations we take linear functions over this domain, so an n-ary symbol f is interpreted by:

$$[\mathrm{f}(\vec{x}_1,\ldots,\vec{x}_n)] = M_1\vec{x}_1 + \ldots + M_n\vec{x}_n + \vec{c} \tag{1}$$

where $\vec{x}_1,\ldots,\vec{x}_n,\vec{c} \in \mathbb{N}^2$ and $M_1,\ldots,M_n \in \mathbb{N}^{2\times 2}$. So an interpretation of a symbol of arity n is given by n square matrices $M_1,\ldots,M_n$ of size 2×2 and one constant vector $\vec{c}$ of dimension 2. Such interpretations are always weakly monotone. We want to use Theorem 3.5a so we need an extended monotone algebra which requires strict monotonicity. For that we need some restrictions and it is easy to see that it can be guaranteed by requiring $M_i[1,1] > 0$ for $1 \le i \le n$.

Now our goal is to prove termination of the given TRS and we will do that by applying Theorem 3.5a instantiated with the extended monotone algebra that we just introduced. We recall that termination is a special case of relative termination so we will apply this theorem with $\mathcal{S} = \mathcal{S}' = \emptyset$. We need to find interpretations for all $f \in \Sigma$. Typically this is done automatically by dedicated tools — we will address this issue in Section 11. One of such tools, TPA [24], applied on this TRS generated the following interpretations:

$$[0] = \begin{pmatrix}0\\2\end{pmatrix} \qquad [\mathrm{fac}(\vec{x})] = \begin{pmatrix}1&2\\0&2\end{pmatrix}\vec{x} + \begin{pmatrix}0\\2\end{pmatrix}$$

$$[\mathrm{p}(\vec{x})] = \begin{pmatrix}1&0\\1&0\end{pmatrix}\vec{x} \qquad [\mathrm{times}(\vec{x},\vec{y})] = \begin{pmatrix}1&0\\0&0\end{pmatrix}\vec{x} + \begin{pmatrix}2&0\\0&0\end{pmatrix}\vec{y}$$

$$[\mathrm{s}(\vec{x})] = \begin{pmatrix}1&1\\3&3\end{pmatrix}\vec{x}$$

Note that the lack of the constant vector $\vec{c}$ in some of the above interpretations indicates that this constant is the zero vector $(0,0)$.

Let us compute interpretations of the left and right hand side of the second rule $\mathrm{fac}(0) \to \mathrm{s}(0)$.

$$[\mathrm{fac}(0)] = \begin{pmatrix}1&2\\0&2\end{pmatrix}\begin{pmatrix}0\\2\end{pmatrix} + \begin{pmatrix}0\\2\end{pmatrix} = \begin{pmatrix}4\\6\end{pmatrix} \qquad [\mathrm{s}(0)] = \begin{pmatrix}1&1\\3&3\end{pmatrix}\begin{pmatrix}0\\2\end{pmatrix} = \begin{pmatrix}2\\6\end{pmatrix}$$

So using our order on vectors we obtain $[\mathrm{fac}(0)] > [\mathrm{s}(0)]$. In a similar way we compute interpretations for the remaining rules. Note that the fact that we restricted ourselves to linear functions means that their composition is linear too and hence all the interpretations that we obtain are of the same shape as in Equation (1).

$$[\mathrm{p}(\mathrm{s}(x))] = \begin{pmatrix}1&0\\1&0\end{pmatrix}\begin{pmatrix}1&1\\3&3\end{pmatrix}\vec{x} = \begin{pmatrix}1&1\\1&1\end{pmatrix}\vec{x}$$

$$[x] = \begin{pmatrix}1&0\\0&1\end{pmatrix}\vec{x}$$

$$[\text{times}(\text{s}(x), \text{fac}(\text{p}(\text{s}(x))))] = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}[\text{s}(x)] + \begin{pmatrix}2 & 0\\ 0 & 0\end{pmatrix}[\text{fac}(\text{p}(\text{s}(x)))] = \begin{pmatrix}7 & 7\\ 0 & 0\end{pmatrix}\vec{x}$$

$$[\text{fac}(\text{s}(x))] = \begin{pmatrix}1 & 2\\ 0 & 2\end{pmatrix}\begin{pmatrix}1 & 1\\ 3 & 3\end{pmatrix}\vec{x} + \begin{pmatrix}0\\ 2\end{pmatrix} = \begin{pmatrix}7 & 7\\ 6 & 6\end{pmatrix}\vec{x} + \begin{pmatrix}0\\ 2\end{pmatrix}$$

For both of the rules it is easy to see that regardless of the assignment to the vector $\vec{x}$ we always obtain that the interpretation of the left hand side is bigger or equal than that of the right hand side of a rule.

All in all we apply Theorem 3.5a with

$$\mathcal{R} = \{\ \text{p}(\text{s}(x)) \to x,\quad \text{fac}(\text{s}(x)) \to \text{times}(\text{s}(x), \text{fac}(\text{p}(\text{s}(x))))\ \}$$
$$\mathcal{R}' = \{\ \text{fac}(0) \to \text{s}(0)\ \}$$
$$\mathcal{S} = \mathcal{S}' = \emptyset$$

This allows us to remove the second rule and conclude termination of the whole system from termination of $\mathcal{R}$ only, which is easy to show, for instance, with the standard method of polynomial interpretations in combination with the dependency pair method. ◁

4 The Arctic Semiring

A *commutative semiring* [18] consists of a carrier D, two designated elements $d_0, d_1 \in D$ and two binary operations $\oplus, \otimes$ on D, called semiring addition and semiring multiplication, respectively, such that both $(D, d_0, \oplus)$ and $(D, d_1, \otimes)$ are commutative monoids and multiplication distributes over addition: $\forall_{x,y,z \in D} : x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$.

One example of a semiring are the natural numbers with the standard operations $\oplus = +$ and $\otimes = *$. We will need the *arctic semiring* (also called the *max/plus algebra*) [15] with carrier $\mathbb{A}_{\mathbb{N}} \equiv \{-\infty\} \cup \mathbb{N}$, where semiring addition is the max operation with neutral element $-\infty$ and semiring multiplication is the standard plus operation with neutral element 0, so:

$$\begin{array}{llll} x \oplus y = y & \text{if } x = -\infty, & x \otimes y = -\infty & \text{if } x = -\infty \text{ or } y = -\infty, \\ x \oplus y = x & \text{if } y = -\infty, & x \otimes y = x + y & \text{otherwise,} \\ x \oplus y = \max(x, y) & \text{otherwise.} & & \end{array}$$

We also consider these operations for arctic numbers *below zero* (*i.e.*, *arctic integers*), that is, on the carrier $\mathbb{A}_{\mathbb{Z}} \equiv \{-\infty\} \cup \mathbb{Z}$.

For any semiring D, we can consider the space of linear functions (square matrices) on n-dimensional vectors over D. These functions (matrices) again form a semiring (though a non-commutative one), and indeed we write $\oplus$ and $\otimes$ for its operations as well.

A semiring is *ordered* [14] by $\geq$ if $\geq$ is a partial order compatible with the operations: $\forall_{x,y,z} : x \geq y \implies x \oplus z \geq y \oplus z$ and $\forall_{x,y,z} : x \geq y \implies x \otimes z \geq y \otimes z$.

The standard semiring of natural numbers is ordered by the standard $\geq$ relation. The semiring of arctic naturals and arctic integers is ordered by $\geq$, being the reflexive closure of $>$ defined as $\ldots > 1 > 0 > -1 > \ldots > -\infty$. Note that standard integers with standard operations form a semiring but it is not ordered in this sense, as we have for instance $1 \geq 0$ but $1 * (-1) = -1 \not\geq 0 = 0 * (-1)$.

We remark that $\geq$ is the "natural" ordering for the arctic semiring, in the following sense: $x \geq y \iff x = x \oplus y$. Since arctic addition is idempotent, some properties of $\geq$ follow easily, like the one presented below.

Lemma 4.1. For arctic integers a_1, a_2, b_1, b_2, if $a_1 \geq a_2 \wedge b_1 \geq b_2$, then $a_1 \oplus b_1 \geq a_2 \oplus b_2$ and $a_1 \otimes b_1 \geq a_2 \otimes b_2$. □

Arctic addition (*i.e.*, the max operation) is not strictly monotone in single arguments: we have, *e.g.*, $5 > 3$ but $5 \oplus 6 = 6 \not> 6 = 3 \oplus 6$. It is, however, "half strict" in the following sense: a strict increase in both arguments simultaneously gives a strict increase in the result, *i.e.*, $a_1 > b_1$ and $a_2 > b_2$ implies $a_1 \oplus a_2 > b_1 \oplus b_2$. There is one exception: arctic addition is obviously strict if one argument is arctic zero, *i.e.*, $-\infty$. This is the motivation for introducing the following relation:

$$a \gg b \iff (a > b) \vee (a = b = -\infty)$$

Below we present some of its properties needed later:

Lemma 4.2. For arctic integers a, a_1, a_2, b_1, b_2,

1. if $a_1 \gg a_2 \wedge b_1 \gg b_2$, then $a_1 \oplus b_1 \gg a_2 \oplus b_2$.
2. if $a_1 \gg a_2 \wedge b_1 \geq b_2$, then $a_1 \otimes b_1 \gg a_2 \otimes b_2$.
3. if $b_1 \gg b_2$, then $a \otimes b_1 \gg a \otimes b_2$.

Proof. By simple case analysis (whether an element is $-\infty$ or not) and some properties of addition and max operations over integers. □

Note that properties 2 and 3 in the above lemma would not hold if we were to replace $\gg$ with $>$.

An arctic natural number $a \in \mathbb{A}_{\mathbb{N}}$ is called *finite* if $a \neq -\infty$. An arctic integer $a \in \mathbb{A}_{\mathbb{Z}}$ is called *positive* if $a \geq 0$ (that excludes $-\infty$ and negative numbers).

Lemma 4.3. Let $m, n \in \mathbb{A}_{\mathbb{N}}$ and $a, b \in \mathbb{A}_{\mathbb{Z}}$, then:

1. if m is finite and n arbitrary, then $m \oplus n$ is finite.
2. if a is positive and b arbitrary, then $a \oplus b$ is positive.
3. if m and n are finite, then $m \otimes n$ is finite.

Proof. Direct computation. □

By analogy to linear algebra over $(\mathbb{N}, +, \cdot)$, we consider sequences (vectors) and rectangular arrays (matrices) of arctic numbers. Sequences $\mathbb{A}^d$ form a semimodule over $\mathbb{A}$, the elements of which we call *arctic vectors*. Operations in the semimodule are $\oplus : \mathbb{A}^d \times \mathbb{A}^d \to \mathbb{A}$ defined by component-wise addition and component-wise multiplication by a scalar value $\otimes : \mathbb{A} \times \mathbb{A}^d \to \mathbb{A}^d$. Then, arctic matrices represent linear functions from vectors to vectors: An arctic matrix M maps a (column) vector $\vec{x}$ to a (column) vector $M \otimes \vec{x}$ and this mapping is linear: $M \otimes (\vec{x} \oplus \vec{y}) = M \otimes \vec{x} \oplus M \otimes \vec{y}$.

We can combine those linear functions (matrices) in the usual way, and we reuse symbols $\oplus$ and $\otimes$ for matrix addition and matrix multiplication. Square arctic matrices form a non-commutative semiring with these operations. *E.g.* the 3×3 identity matrix is

$$\begin{pmatrix} 0 & -\infty & -\infty \\ -\infty & 0 & -\infty \\ -\infty & -\infty & 0 \end{pmatrix}$$

We will be interested in linear functions over arctic vectors of the following shape:

Definition 4.4. Let $\mathbb{A}$ be an arctic domain (so either arctic naturals $\mathbb{A}_\mathbb{N}$ or arctic integers $\mathbb{A}_\mathbb{Z}$). An *(n-ary) arctic linear function (over $\mathbb{A}$)* (with linear factors $M_1, \ldots, M_n$ and an absolute part $\vec{c}$) is a function of the following shape:

$$f(\vec{x}_1, \ldots, \vec{x}_n) = M_1 \otimes \vec{x}_1 \oplus \ldots \oplus M_n \otimes \vec{x}_n \oplus \vec{c}$$

So an arctic linear function over column vectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{A}^d$ is described by a column vector $\vec{c} \in \mathbb{A}^d$ and square matrices $M_1, \ldots, M_n \in \mathbb{A}^{d \times d}$. ◇

Note that for brevity from now on we will omit the semiring multiplication sign $\otimes$ and use the following notation for arctic linear functions:

$$f(\vec{x}_1, \ldots, \vec{x}_n) = M_1 \vec{x}_1 \oplus \ldots \oplus M_n \vec{x}_n \oplus \vec{c}$$

Example 4.5. Consider a linear function:

$$f(\vec{x}, \vec{y}) = \begin{pmatrix} 1 & -\infty \\ 0 & -\infty \end{pmatrix} \vec{x} \oplus \begin{pmatrix} -\infty & -\infty \\ 0 & 1 \end{pmatrix} \vec{y} \oplus \begin{pmatrix} -\infty \\ 0 \end{pmatrix}$$

Evaluation of this function on some exemplary arguments yields:

$$f(\begin{pmatrix} -\infty \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -\infty \end{pmatrix}) = \begin{pmatrix} 1 & -\infty \\ 0 & -\infty \end{pmatrix} \begin{pmatrix} -\infty \\ 0 \end{pmatrix} \oplus \begin{pmatrix} -\infty & -\infty \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -\infty \end{pmatrix} \oplus \begin{pmatrix} -\infty \\ 0 \end{pmatrix} = \begin{pmatrix} -\infty \\ 1 \end{pmatrix}$$

◁

5 Weighted Tree Automata

In this section we instantiate the monotone algebra framework with the initial algebraic semantics of weighted tree automata of a certain shape. This allows to

put the matrix method (Example 3.6) into perspective, and it also is the basis for the generalization to arctic matrices (following sections).

A weighted tree automaton [10, 9] is a finite-state device that computes a mapping from trees over some signature into some semiring. This computational model is obtained from classical (Boolean) automata by assigning weights to transitions.

Formally, a D-weighted tree automaton is a tuple $A = (D, Q, \Sigma, \delta, F)$ where D is a semiring, Q is a finite set of states, Σ is a ranked signature, δ is a transition function that assigns to any k-ary symbol $f \in \Sigma_k$ a function $\delta_f : Q^k \times Q \to D$ and F is a mapping $Q \to D$. The idea is that $\delta_f(q_1, \ldots, q_k, q)$ gives the weight of the transition from $(q_1, \ldots, q_k)$ to q, and $F(q)$ gives the weight of the final state q.

We use the following tree automaton as an ongoing example for this section. This example is related to the matrix interpretation shown in Example 3.6, in a way that will be made precise later.

Example 5.1. For the signature $\Sigma = \{0/0, \mathrm{p}/1, \mathrm{s}/1, \mathrm{fac}/1, \mathrm{times}/2\}$ (from Example 2.3), a $\mathbb{N}$-weighted tree automaton with states $Q = \{a, b, c\}$ is given by:

$$\begin{aligned}
&(0) && \delta_0(b) = 2, \delta_0(c) = 1,\\
&(\mathrm{p}) && \delta_{\mathrm{p}}(a,a) = \delta_{\mathrm{p}}(a,b) = \delta_{\mathrm{p}}(c,c) = 1,\\
&(\mathrm{s}) && \delta_{\mathrm{s}}(a,a) = \delta_{\mathrm{s}}(b,a) = 1,\ \delta_{\mathrm{s}}(a,b) = \delta_{\mathrm{s}}(b,b) = 3,\ \delta_{\mathrm{s}}(c,c) = 1,\\
&(\mathrm{fac}) && \delta_{\mathrm{fac}}(a,a) = 1,\ \delta_{\mathrm{fac}}(b,a) = \delta_{\mathrm{fac}}(b,b) = \delta_{\mathrm{fac}}(c,b) = 2,\ \delta_{\mathrm{fac}}(c,c) = 1,\\
&(\mathrm{times}) && \delta_{\mathrm{times}}(a,c,a) = 1,\ \delta_{\mathrm{times}}(c,b,a) = 2,\ \delta_{\mathrm{times}}(c,c,c) = 1,
\end{aligned}$$

$F(a) = 1$, and all other transitions have weight 0. ◁

For any tree $t = f(t_1, \ldots, t_k)$ over Σ, and $q \in Q$, denote by $A_q(t)$ the weight that A assigns to t in state q:

$$A_q(t) = \sum\{\delta_f(q_1, \ldots, q_k, q) \cdot A_{q_1}(t_1) \cdot \ldots \cdot A_{q_k}(t_k) \mid q_1, \ldots, q_k, q \in Q\}$$

and the total weight $A(t)$ is $\sum\{F(q) \cdot A_q(t) \mid q \in Q\}$.

Example 5.2. (continued) We find $A_a(0) = 0, A_b(0) = 2, A_c(0) = 1$, since the symbol 0 is nullary and thus $A_q(0) = \delta_0(q)$. Then, for example, $A_b(\mathrm{s}(0)) = \delta_{\mathrm{s}}(a,b) \cdot A_a(0) + \delta_{\mathrm{s}}(b,b) \cdot A_b(0) + \delta_{\mathrm{s}}(c,b) \cdot A_c(0) = 3 \cdot 0 + 3 \cdot 2 + 0 \cdot 1 = 6$. ◁

This is called *initial algebra semantics* of a tree automaton. Indeed, the automaton is a Σ-algebra where the carrier set consists of weight vectors, indexed by states. Let $V = (Q \to D)$ be the set of such vectors. Then for each k-ary symbol f, the transition δ_f computes a function $[\delta_f] : V^k \to V$ by $[\delta_f](v_1, \ldots, v_k) = w$ where

$$w_q = \sum\{\delta_f(q_1, \ldots, q_k, q) \cdot v_{1,q_1} \cdot \ldots \cdot v_{k,q_k} \mid q_1, \ldots, q_k \in Q\}.$$

Example 5.3. (continued) For the unary fac symbol, we have the unary function

$$[\mathrm{fac}] : V^1 \to V : (v_{1,a}, v_{1,b}, v_{1,c}) \mapsto (v_{1,a} + 2v_{1,b},\ 2v_{1,b} + 2v_{1,c},\ v_{1,c}).$$

Since 0 is a nullary symbol, its interpretation $[0]$ is of type $V^0 \to V$, that is, it takes an empty argument list and produces a vector $[0] = (0, 2, 1)$. ◁

By distributivity in the semiring, each function $[\delta_f]$ is multilinear (linear in each argument):

$$\begin{aligned} & [\delta_f](\ldots, v_{i-1}, v_i + v_i', v_{i+1}, \ldots) \\ = \; & [\delta_f](\ldots, v_{i-1}, v_i, v_{i+1}, \ldots) + [\delta_f](\ldots, v_{i-1}, v_i', v_{i+1}, \ldots). \end{aligned}$$

For a given tree automaton A over Σ, the collection $\{[\delta_f] \mid f \in \Sigma\}$ constitutes an algebra with carrier V. Therefore, interpretations of function symbols $[\delta_f]$ can be lifted to interpretations of terms.

Example 5.4. (continued) In the algebra of the automaton:

$$[\mathrm{fac}(0)] = (0 + 2 \cdot 2,\ 2 \cdot 2 + 2 \cdot 1,\ 1) = (4, 6, 1). \qquad \triangleleft$$

It is convenient to think of elements of V as column vectors, and F as a row vector. Then $A(t)$ is the dot product $F \cdot (A_1(t), \ldots, A_{|Q|}(t))^T$.

Example 5.5. (continued) $A(\mathrm{fac}(0)) = (1, 0, 0) \cdot (4, 6, 1)^T = 4.$ $\triangleleft$

With these preparations, we can apply the monotone algebra approach for proving termination of term rewriting, where the algebra is given by a finite weighted tree automaton.

In order to obtain a method that can be automated easily, we restrict the shape of the automata transitions, so that the interpretation of each function symbol is a sum of linear functions in single arguments, and an absolute part, cf. Equation 1.

Definition 5.6. A weighted tree automaton $A = (D, Q, \Sigma, \delta, F)$ is called *path-separated* with initial state $i \in Q$ if for each k-ary transition with non-zero weight we have that

- at most one of the initial k arguments is $\neq i$:

$$\delta_f(q_1, \ldots, q_k, q) \neq 0 \Rightarrow \exists_{\leq 1} 1 \leq j \leq k : q_j \neq i.$$

- if the target is i, then all sources are i, and the weight is unit:

$$\delta_f(q_1, \ldots, q_k, i) = (\text{if } q_1 = \ldots = q_k = i \text{ then } 1 \text{ else } 0). \qquad \diamond$$

Example 5.7. (continued) The given automaton is path-separated, with $i = c$ as the initial state. $\triangleleft$

Proposition 5.8. The following conditions are equivalent for a weighted tree automaton $A = (D, Q, \Sigma, \delta, F)$:

- A is path-separated with initial state i,
- each action of $[\delta_f]$ has the following form:

$$[\delta_f](\vec{v}_1, \ldots, \vec{v}_k) \mapsto M_1 \cdot \vec{v}_1 + \ldots + M_k \cdot \vec{v}_k + \vec{a},$$

where M_j are square matrices of dimension $|Q| \times |Q|$, with all entries in row i and in column i are zero; and $\vec{a}$ is a vector, with entry one at position i.

Proof. Let A be path-separated with an initial state i. For any $f \in \Sigma_k$, we have $\vec{a}[q] = \delta_f(i,\ldots,i,q)$, if none of the first k arguments is $\neq i$, and $M_j[q,p] = \delta_f(i,\ldots,i,p,i,\ldots,i,q)$ where p is the single non-i state among the first k arguments. By the path-separation restriction, these cases cover all possible transitions. □

Example 5.9. (continued) $[\mathrm{fac}](\vec{v}) = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \vec{v} + \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}$. ◁

Under these conditions, for each t we have $A_i(t) = 1$. So we drop the entry at i in $\vec{a}$, and also each row i and each column i in M_j. Then by Proposition 5.8, a path-separated tree automaton corresponds to a matrix interpretation of shape (1) and vice-versa.

Example 5.10. (continued) $[\mathrm{fac}](\vec{v}) = \begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix} \vec{v} + \begin{pmatrix} 0 \\ 2 \end{pmatrix}$. ◁

We call these tree automata path-separated because their semantics can be computed as the sum of matrix products along all paths of the input, and the values along different paths do not influence each other.

Here, a *path* is a sequence of function symbols with directions. Formally, for any term $t = f(t_1,\ldots,t_k)$, we define

$$\mathrm{paths}(t) = \{f_0\} \cup \{f_i \circ p \mid 1 \le i \le k, p \in \mathrm{paths}(t_i)\}.$$

This is a mapping from $\mathcal{T}(\Sigma)$ to nonempty sequences of pairs of symbols and numbers, with a pair (f,i) denoted by f_i; actually to a subset of $P_\Sigma = (\Sigma \times \mathbb{N}_{>0})^*(\Sigma \times \{0\})$.

Example 5.11.

$$\mathrm{paths}(\mathrm{times}(0,\mathrm{fac}(0))) = \{\mathrm{times}_0,\ \mathrm{times}_1 \circ 0_0,\ \mathrm{times}_2 \circ \mathrm{fac}_0,\ \mathrm{times}_2 \circ \mathrm{fac}_1 \circ 0_0\}.$$ ◁

For a path-separated tree automaton $A = (D,Q,\Sigma,\delta,F)$ and each k-ary symbol f, where δ_f is as in (1), define a mapping $[\cdot]$ from paths in P_Σ to vectors by $[f_0] = \vec{c}$ and $[f_i \circ p] = M_i \cdot [p]$. Then it follows from distributivity of addition (of vectors) over multiplication (with matrices) that for each term t,

$$A(t) = \sum\{F \cdot [p] \mid p \in \mathrm{paths}(t)\}.$$

This illustrates why we call these automata path-separated.

We briefly comment on the effect of the path-separation restriction. Consider a signature with a binary symbol g. A matrix interpretation of dimension one interprets g with a function $(x_1,x_2) \mapsto m_1x_1 + m_2x_2 + a$. This corresponds to a path-separated $\mathbb{N}$-weighted automaton with just two states, one of which being the initial state.

The general form of a transition function of a tree automaton with two states, one of them initial, is $(x_1,x_2) \mapsto m_{12}x_1x_2 + m_1x_1 + m_2x_2 + a$. The "$m_{12}x_1x_2$"

component cannot be part of a path-separated tree automaton's transition function. We really lose expressiveness here, *e.g.*, the tree automaton's transition $(x_1, x_2) \mapsto x_1 x_2$ cannot be expressed by matrix interpretations, even with additional states, since it grows faster (doubly exponential) than any matrix-representable function (exponential).

On the other hand, if the signature contains no symbols of arity > 1, then each tree automaton has an equivalent path-separated automaton (of size $|Q| + 1$, since in general we need to add the initial state).

6 Full Arctic Termination

In this section, we instantiate the monotone algebra approach for proving termination of rewriting by using algebras defined by path-separated arctic tree automata.

The algebra domain consists of vectors of arctic naturals, $\mathbb{A}_{\mathbb{N}}^d$. Every $f \in \Sigma$ will be interpreted by an arctic linear function (Definition 4.4) and we will refer to such interpretations as *arctic* Σ*-interpretations*.

We define orders on arctic vectors and matrices by taking a point-wise extension of the orders $\gg$ and $\geq$ introduced in Section 4. We will use the same notation, *i.e.*, $\gg$ and $\geq$, for those lifted orders. Now we take the vector extension of $\gg$ and $\geq$ as, respectively, the strict and non-strict order of the algebra. Note that they are compatible, *i.e.*, $\gg \cdot \geq \subseteq \gg$. However with this choice we do not get well-foundedness of the strict order as $-\infty \gg -\infty$. Therefore we will restrict first components of vectors to finite elements (*i.e.*, elements different from $-\infty$, as introduced before Lemma 4.3). Effectively our algebra becomes $(\mathbb{N} \times \mathbb{A}_{\mathbb{N}}^{d-1}, \{f_A\}_{f \in \Sigma}, \gg, \geq)$.

We will consider arctic linear functions over the domain of our algebra, so we must make sure that evaluation of those functions stays within the domain, *i.e.*, that the first vector component is finite. The following definition and lemma address this issue.

Definition 6.1. An n-ary arctic linear function

$$f(\vec{x}_1, \ldots, \vec{x}_n) = M_1\vec{x}_1 \oplus \ldots \oplus M_n\vec{x}_n \oplus \vec{c}$$

over $\mathbb{A}_{\mathbb{N}}$ is called *somewhere finite* if:

- $\vec{c}[1]$ is finite, or
- $M_i[1,1]$ is finite for some $1 \leq i \leq n$. ◇

Lemma 6.2. Let f be an n-ary arctic linear function over $\mathbb{A}_{\mathbb{N}}$, $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{N} \times \mathbb{A}_{\mathbb{N}}^{d-1}$ and $\vec{v} = f(\vec{x}_1, \ldots, \vec{x}_n)$. If f is somewhere finite then $\vec{v}[1]$ is finite.

Proof.

$$f(\vec{x}_1, \ldots, \vec{x}_n)[1] = (M_1\vec{x}_1)[1] \oplus \ldots \oplus (M_n\vec{x}_n)[1] \oplus \vec{c}[1] \tag{2}$$

Since f is somewhere finite we have:

- $\vec{c}[1]$ is finite, or

- for some $1 \leq i \leq n$, $M_i[1,1]$ is finite but then $(M_i\vec{x}_i)[1] = M_i[1,1]\vec{x}_i[1] \oplus \ldots \oplus M_i[1,d]\vec{x}_i[d]$, which is finite by Lemma 4.3, as $M_i[1,1]$ is finite.

In either case one of the summands in Equation 2 is finite, making the whole expression finite by Lemma 4.3. □

To apply the monotone algebra theorem, Theorem 3.5, we will need to compare arctic linear functions, *i.e.*, we will need some properties ensuring that, for arbitrary arguments, one arctic function always gives a vector that is greater (or greater equal) than the result of application of some other arctic functions to the same arguments. This is addressed in the following lemma, which is the arctic counterpart of the absolute positiveness criterion used for polynomial interpretations [23].

Definition 6.3. Let f, g be arctic linear functions over $\mathbb{A}$:

$$f(\vec{x}_1, \ldots, \vec{x}_n) = M_1\vec{x}_1 \oplus \ldots \oplus M_n\vec{x}_n \oplus \vec{c}$$
$$g(\vec{x}_1, \ldots, \vec{x}_n) = N_1\vec{x}_1 \oplus \ldots \oplus N_n\vec{x}_n \oplus \vec{d}$$

We will say that f is greater (resp. greater equal) than g, notation $f \gg_\lambda g$ (resp. $f \geq_\lambda g$) iff:

- $c \gg d$ (resp. $c \geq d$) and
- $\forall_{1 \leq i \leq n} : M_i \gg N_i$ (resp. $M_i \geq N_i$). ◇

We will justify the above definition in Lemma 6.5, but first we need an auxiliary result:

Lemma 6.4. Let $M, N \in \mathbb{A}^{d \times d}$ and $\vec{x}, \vec{y} \in \mathbb{A}^d$.

1. If $M \gg N$ and $\vec{x} \geq \vec{y}$ then $M\vec{x} \gg N\vec{y}$.
2. If $M \geq N$ and $\vec{x} \geq \vec{y}$ then $M\vec{x} \geq N\vec{y}$.

Proof. Immediate using Lemma 4.1 and the first two properties of Lemma 4.2. □

Lemma 6.5. Let f, g be arctic linear functions over $\mathbb{A}$ and let $\vec{x}_1, \ldots, \vec{x}_n$ be arbitrary vectors.

1. If $f \gg_\lambda g$ then $f(\vec{x}_1, \ldots, \vec{x}_n) \gg g(\vec{x}_1, \ldots, \vec{x}_n)$.
2. If $f \geq_\lambda g$ then $f(\vec{x}_1, \ldots, \vec{x}_n) \geq g(\vec{x}_1, \ldots, \vec{x}_n)$.

Proof. We will prove only the first case — the other one is analogous.

$$f(\vec{x}_1, \ldots, \vec{x}_n) = M_1\vec{x}_1 \oplus \ldots \oplus M_n\vec{x}_n \oplus \vec{c}$$
$$g(\vec{x}_1, \ldots, \vec{x}_n) = N_1\vec{x}_1 \oplus \ldots \oplus N_n\vec{x}_n \oplus \vec{d}$$

We have $\vec{c} \gg \vec{d}$ and $\forall_{1 \leq i \leq n} : M_i \gg N_i$ as $f \gg_\lambda g$ and hence $M_i\vec{x}_i \gg N_i\vec{x}_i$ by Lemma 6.4. So every vector summand of the evaluation of f is related by $\gg$ with a corresponding summand of g and we conclude by Lemma 4.1. □

Clearly arctic linear functions are weakly monotone (because so is the max operation, *i.e.*, arctic addition) and we establish this property in the following lemma.

Lemma 6.6. Every arctic linear function f over $\mathbb{A}$ is monotone with respect to $\geq$.

Proof. Let $x_i \geq x_i'$. We have:

$$f(\vec{x}_1, \ldots, \vec{x}_i, \ldots, \vec{x}_n) = M_1\vec{x}_1 \oplus \ldots \oplus M_i\vec{x}_i \oplus \ldots \oplus M_n\vec{x}_n \oplus \vec{c}$$
$$f(\vec{x}_1, \ldots, \vec{x}_i', \ldots, \vec{x}_n) = M_1\vec{x}_1 \oplus \ldots \oplus M_i\vec{x}_i' \oplus \ldots \oplus M_n\vec{x}_n \oplus \vec{c}$$

All the summands are equal except for the one corresponding to the i'th argument, where we have $M_i\vec{x}_i \geq M_i\vec{x}_i'$ by Lemma 6.4 and we conclude

$$f(\vec{x}_1, \ldots, \vec{x}_i, \ldots, \vec{x}_n) \geq f(\vec{x}_1, \ldots, \vec{x}_i', \ldots, \vec{x}_n)$$

by Lemma 4.1. □

However, to obtain an extended weakly monotone algebra, and prove full termination using it, we need strict monotonicity. As remarked in Section 4, arctic addition is not strictly monotone. Hence functions introduced in Definition 4.4 are strictly monotone only if the $\oplus$ operation is essentially redundant; for instance it is immediately lost for functions of more than one argument. This essentially restricts our method to unary rewriting [35]; a proper extension of string rewriting. As such, it had been described in [37] and had been applied by Matchbox in the 2007 termination competition. The following theorem provides a termination criterion for such systems. In the next section we will look at top termination problems, which will allow us to lift this restriction and consider arbitrary TRSs.

Theorem 6.7. Let $\mathcal{R}, \mathcal{R}', \mathcal{S}, \mathcal{S}'$ be TRSs over a signature Σ and $[\cdot]$ be an arctic Σ-interpretation over $\mathbb{A}_{\mathbb{N}}$. If:

- every function symbol has arity at most 1,
- every constant $a \in \Sigma$ is interpreted by $[a] = \vec{c}$ with $\vec{c}[1]$ finite,
- every unary symbol $s \in \Sigma$ is interpreted by $[s(\vec{x})] = M \otimes \vec{x}$ with $M[1,1]$ finite,
- $[\ell] \geq_\lambda [r]$ for every rule $\ell \to r \in \mathcal{R} \cup \mathcal{S}$,
- $[\ell] \gg_\lambda [r]$ for every rule $\ell \to r \in \mathcal{R}' \cup \mathcal{S}'$ and
- $\mathrm{SN}(\mathcal{R}/\mathcal{S})$.

Then $\mathrm{SN}(\mathcal{R} \cup \mathcal{R}'/\mathcal{S} \cup \mathcal{S}')$.

Proof. By Theorem 3.5a. Note that, by Lemma 6.5, $[\ell] \geq_\lambda [r]$ (resp. $[\ell] \gg_\lambda [r]$) implies $[\ell] \geq_\alpha [r]$ (resp. $[\ell] \gg_\alpha [r]$). So we only need to show that $(\mathbb{N} \times \mathbb{A}_{\mathbb{N}}^{d-1}, [\cdot], \gg, \geq)$ is an extended monotone algebra. The order $\gg$ is well-founded on this domain as with every decrease we get a decrease in the first component of the vector, which

belongs to $\mathbb{N}$. Arctic functions are always weakly monotone by Lemma 6.6 and it is an easy observation that, due to the first three premises of this theorem, the interpretations that we allow here are strictly monotone. Finally we stay within the domain by Lemma 6.2 as the interpretation functions $[f]$ that we restrict to are somewhere finite (again by the first three assumptions). □

We now present an example illustrating this theorem.

Example 6.8. The relative termination problem `SRS/Waldmann/r2` is

$$\{\mathtt{c\,a\,c} \to \epsilon,\ \mathtt{a\,c\,a} \to \mathtt{a}^4\ /\ \epsilon \to \mathtt{c}^4\}$$

In the 2007 termination competition, it had been solved by Jambox [12] via "self labeling" and by Matchbox via essentially the following arctic proof.

We use the following arctic interpretation

$$[\mathtt{a}](\vec{x}) = \begin{pmatrix} 0 & 0 & -\infty \\ 0 & 0 & -\infty \\ 1 & 1 & 0 \end{pmatrix} \vec{x} \qquad [\mathtt{c}](\vec{x}) = \begin{pmatrix} 0 & -\infty & -\infty \\ -\infty & -\infty & 0 \\ -\infty & 0 & -\infty \end{pmatrix} \vec{x}$$

It is immediate that [c] is a permutation (it swaps the second and third component of its argument vector), so $[\mathtt{c}]^2 = [\mathtt{c}]^4$ is the identity and we have $[\epsilon] = [\mathtt{c}]^4$. A short calculation shows that [a] is idempotent, so $[\mathtt{a}] = [\mathtt{a}^4]$. We compute

$$[\mathtt{c\,a\,c}](\vec{x}) = \begin{pmatrix} 0 & -\infty & 0 \\ 1 & 0 & 1 \\ 0 & -\infty & 0 \end{pmatrix} \vec{x} \quad [\mathtt{a\,c\,a}](\vec{x}) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 2 & 2 & 1 \end{pmatrix} \vec{x} \quad [\mathtt{a}^4](\vec{x}) = \begin{pmatrix} 0 & 0 & -\infty \\ 0 & 0 & -\infty \\ 1 & 1 & 0 \end{pmatrix} \vec{x}$$

therefore $[\mathtt{c\,a\,c}](\vec{x}) \geq_\lambda [\epsilon](\vec{x})$ and $[\mathtt{a\,c\,a}](\vec{x}) \gg_\lambda [\mathtt{a}^4](\vec{x})$. Note also that all the top left entries of matrices are finite. This allows us to remove the strict rule $\mathtt{a\,c\,a} \to \mathtt{a}^4$ using Theorem 6.7. The remaining strict rule can be removed by counting letters a. ◁

7 Arctic Top Termination

As explained earlier, there are no strictly monotone, linear arctic functions of more than one argument. Therefore in this section we change our attention from full termination to top termination problems, where only weak monotonicity is required. This is not a very severe restriction as it fits with the widely used dependency pair method that replaces a full termination problem with an equivalent top termination problem, as remarked in Section 2.

The monotone algebra that we are going to use is the same as in Section 6, *i.e.*, $(\mathbb{N} \times \mathbb{A}_{\mathbb{N}}^{d-1}, \{f_A\}_{f \in \Sigma}, \gg, \geq)$. However now for proving top termination we will employ the second part of Theorem 3.5, so we only need a monotone algebra, instead of an extended monotone algebra. This allows us to consider arbitrary TRSs, as without the requirement of strict monotonicity we can allow arctic linear functions of more than one argument. The following theorem allows us to prove top termination in this setting:

Theorem 7.1. Let $\mathcal{R}, \mathcal{R}', \mathcal{S}$ be TRSs over a signature Σ and $[\cdot]$ be an arctic Σ-interpretation over $\mathbb{A}_{\mathbb{N}}$. If:

- for each $f \in \Sigma$, $[f]$ is somewhere finite,
- $[\ell] \geq_\lambda [r]$ for every rule $\ell \to r \in \mathcal{R} \cup \mathcal{S}$,
- $[\ell] \gg_\lambda [r]$ for every rule $\ell \to r \in \mathcal{R}'$ and
- $\mathrm{SN}(\mathcal{R}_{\mathrm{top}}/\mathcal{S})$.

Then $\mathrm{SN}(\mathcal{R}_{\mathrm{top}} \cup \mathcal{R}'_{\mathrm{top}}/\mathcal{S})$.

Proof. By Theorem 3.5b. By the same argument as in Theorem 6.7, $(\mathbb{N} \times \mathbb{A}_{\mathbb{N}}^{d-1}, [\cdot], \gg, \geq)$ is a weakly monotone algebra. So we only need to show that the evaluation stays within the algebra domain which follows from Lemma 6.2 and the first assumption. □

We will illustrate this theorem on an example now.

Example 7.2. Consider the rewriting system `secret05/tpa2`:

$$\begin{array}{ll} (1)\ \mathrm{f}(\mathrm{s}(x), y) \to \mathrm{f}(\mathrm{p}(\mathrm{s}(x) - y), \mathrm{p}(y - \mathrm{s}(x))) & (3)\quad \mathrm{p}(\mathrm{s}(x)) \to x \\ (2)\ \mathrm{f}(x, \mathrm{s}(y)) \to \mathrm{f}(\mathrm{p}(x - \mathrm{s}(y)), \mathrm{p}(\mathrm{s}(y) - x)) & (4)\quad x - 0 \to x \\ & (5)\ \mathrm{s}(x) - \mathrm{s}(y) \to x - y \end{array}$$

It was solved in the 2007 competition by AProVE [16] using narrowing followed by polynomial interpretations and by TTT2 [28] using polynomial interpretations with negative constants. In 2008 both provers used arctic interpretations to solve this problem.

After the dependency pair transformation, 9 dependency pairs can be removed using polynomial interpretations leaving the essential two dependency pairs:

$$\begin{aligned} (1^\sharp)\ & \mathrm{f}^\sharp(\mathrm{s}(x), y) \to \mathrm{f}^\sharp(\mathrm{p}(\mathrm{s}(x) - y), \mathrm{p}(y - \mathrm{s}(x))) \\ (2^\sharp)\ & \mathrm{f}^\sharp(x, \mathrm{s}(y)) \to \mathrm{f}^\sharp(\mathrm{p}(x - \mathrm{s}(y)), \mathrm{p}(\mathrm{s}(y) - x)) \end{aligned}$$

So now, according to the dependency pair Theorem 2.2, we need to consider the relative top termination problem $\mathrm{SN}(\mathcal{R}_{\mathrm{top}}/\mathcal{S})$, where $\mathcal{R} = \{(1^\sharp), (2^\sharp)\}$ and $\mathcal{S} = \{(1), (2), (3), (4), (5)\}$. For that consider the following arctic interpretation

$$[\mathrm{f}^\sharp(\vec{x}, \vec{y})] = \begin{pmatrix} -\infty & -\infty \\ -\infty & -\infty \end{pmatrix} \vec{x} \oplus \begin{pmatrix} 0 & 0 \\ -\infty & -\infty \end{pmatrix} \vec{y} \oplus \begin{pmatrix} 0 \\ -\infty \end{pmatrix} \qquad [0] = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

$$[\vec{x} - \vec{y}] = \begin{pmatrix} 0 & -\infty \\ 0 & 0 \end{pmatrix} \vec{x} \oplus \begin{pmatrix} -\infty & -\infty \\ 0 & 0 \end{pmatrix} \vec{y} \oplus \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad [\mathrm{p}(\vec{x})] = \begin{pmatrix} 0 & -\infty \\ 0 & -\infty \end{pmatrix} \vec{x} \oplus \begin{pmatrix} -\infty \\ -\infty \end{pmatrix}$$

$$[\mathrm{f}(\vec{x}, \vec{y})] = \begin{pmatrix} 0 & 0 \\ 0 & -\infty \end{pmatrix} \vec{x} \oplus \begin{pmatrix} 2 & 0 \\ 0 & -\infty \end{pmatrix} \vec{y} \oplus \begin{pmatrix} 0 \\ -\infty \end{pmatrix} \qquad [\mathrm{s}(\vec{x})] = \begin{pmatrix} 0 & 0 \\ 2 & 1 \end{pmatrix} \vec{x} \oplus \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

which is somewhere finite and removes the second dependency pair:

$$[\mathrm{f}^\sharp(x, \mathrm{s}(y))] = \begin{pmatrix} -\infty & -\infty \\ -\infty & -\infty \end{pmatrix} \vec{x} \oplus \begin{pmatrix} 2 & 1 \\ -\infty & -\infty \end{pmatrix} \vec{y} \oplus \begin{pmatrix} 2 \\ -\infty \end{pmatrix}$$
$$[\mathrm{f}^\sharp(\mathrm{p}(x - \mathrm{s}(y)), \mathrm{p}(\mathrm{s}(y) - x))] = \begin{pmatrix} -\infty & -\infty \\ -\infty & -\infty \end{pmatrix} \vec{x} \oplus \begin{pmatrix} 0 & 0 \\ -\infty & -\infty \end{pmatrix} \vec{y} \oplus \begin{pmatrix} 0 \\ -\infty \end{pmatrix}$$

It is also weakly compatible with all the rules. The remaining dependency pair can be removed by a standard matrix interpretation of dimension two. ◁

8 ... Below Zero

In this section we will boldly go below zero: we extend the domain of matrix and vector coefficients from $\mathbb{A}_\mathbb{N}$ (arctic naturals) to $\mathbb{A}_\mathbb{Z}$ (arctic integers). This allows to interpret some function symbols by the "predecessor" function $x \mapsto x - 1$, and so represents their "intrinsic" semantics. This is the same motivation as the one for allowing polynomial interpretations with negative coefficients [19, 20].

We need to be careful though, as the relation $\gg$ on vectors of arctic integers is not well-founded. We will solve it in a similar way as in Sections 6 and 7, that is by restricting the first component of the vectors in our domain to natural numbers, which restores well-foundedness. So we are working in the $(\mathbb{N} \times \mathbb{A}_\mathbb{Z}^{d-1}, \{f_A\}_{f \in \Sigma}, \gg, \geq)$ algebra.

Again we need to make sure that we do not go outside of the domain, *i.e.*, the first vector component needs to be positive. This is ensured by the following property:

Definition 8.1. An n-ary arctic linear function

$$f(\vec{x}_1, \ldots, \vec{x}_n) = M_1 \otimes \vec{x}_1 \oplus \ldots \oplus M_n \otimes \vec{x}_n \oplus \vec{c}$$

over $\mathbb{A}_\mathbb{Z}$ is called *absolutely positive* if $\vec{c}[1]$ is positive. ◇

Lemma 8.2. Let f be an n-ary arctic linear function over $\mathbb{A}_\mathbb{Z}$, $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{N} \times \mathbb{A}_\mathbb{Z}^{d-1}$ and $\vec{v} = f(\vec{x}_1, \ldots, \vec{x}_n)$. If f is absolutely positive then $\vec{v}[1] \in \mathbb{N}$.

Proof. Immediate, as $\vec{c}[1]$ positive by the definition of absolutely positive function.

$$\vec{v}[1] = f(\vec{x}_1, \ldots, \vec{x}_n)[1] = \max(\vec{c}[1], \ldots) \geq 0$$ □

We can now present the main theorem of this section.

Theorem 8.3. Let $\mathcal{R}, \mathcal{R}', \mathcal{S}$ be TRSs over a signature Σ and $[\cdot]$ be an arctic Σ-interpretation over $\mathbb{A}_\mathbb{Z}$. If:

- for each $f \in \Sigma$, $[f]$ is absolutely positive,
- $[\ell] \geq_\lambda [r]$ for every rule $\ell \to r \in \mathcal{R} \cup \mathcal{S}$,

- $[\ell] \gg_\lambda [r]$ for every rule $\ell \to r \in \mathcal{R}'$ and
- $\mathrm{SN}(\mathcal{R}_{\mathrm{top}}/\mathcal{S})$.

Then $\mathrm{SN}(\mathcal{R}_{\mathrm{top}} \cup \mathcal{R}'_{\mathrm{top}}/\mathcal{S})$.

Proof. By Theorem 3.5b. We proved that $(\mathbb{N} \times \mathbb{A}_{\mathbb{N}}^{d-1}, \{f_A\}_{f\in\Sigma}, \gg, \geq)$ is a weakly monotone algebra in Theorem 7.1 — now the domain is extended from arctic naturals to arctic integers but all the properties carry over easily. The fact that we respect the algebra domain is ensured by the first property and Lemma 8.2. □

We now illustrate this theorem on an example.

Example 8.4. Let us consider the Beerendonk/2.trs TRS from the TPDB [2], consisting of the following six rules:

$$\begin{aligned} \mathrm{cond}(\mathrm{true}, x, y) &\to \mathrm{cond}(\mathrm{gr}(x, y), \mathrm{p}(x), \mathrm{s}(y)) & \mathrm{gr}(\mathrm{s}(x), \mathrm{s}(y)) &\to \mathrm{gr}(x, y) \\ \mathrm{gr}(0, x) &\to \mathrm{false} & \mathrm{gr}(\mathrm{s}(x), 0) &\to \mathrm{true} \\ \mathrm{p}(0) &\to 0 & \mathrm{p}(\mathrm{s}(x)) &\to x \end{aligned}$$

This is a straightforward encoding of the following imperative program

```
while x > y do (x, y) := (x-1, y+1);
```

with $x, y \in \mathbb{N}$ and the predecessor of x, *i.e.*, $x - 1$, defined on this domain, so $0 - 1 = 0$. This program is obviously terminating, however its encoding as the above TRS posed a serious challenge for the tools in the termination competition. We will now show a termination proof for this system using an arctic below zero interpretation.

We begin by applying the dependency pair method and obtaining four dependency pairs, three of which can be easily removed (for instance using standard matrix or polynomial interpretations) leaving the following single dependency pair:

$$\mathrm{cond}^\sharp(\mathrm{true}, x, y) \to \mathrm{cond}^\sharp(\mathrm{gr}(x, y), \mathrm{p}(x), \mathrm{s}(y))$$

Now, consider the following arctic matrix interpretation of dimension 1, so a degenerated case where arctic vectors and matrices simply become arctic numbers:

$$\begin{aligned} [\mathrm{cond}^\sharp(\vec{x}, \vec{y}, \vec{z})] &= (0)\vec{x} \oplus (0)\vec{y} \oplus (-\infty)\vec{z} \oplus (0) & [0] &= (0) \\ [\mathrm{cond}(\vec{x}, \vec{y}, \vec{z})] &= (0)\vec{x} \oplus (2)\vec{y} \oplus (-\infty)\vec{z} \oplus (0) & [\mathrm{false}] &= (0) \\ [\mathrm{gr}(\vec{x}, \vec{y})] &= (-1)\vec{x} \oplus (-\infty)\vec{y} \oplus (0) & [\mathrm{true}] &= (2) \\ [\mathrm{p}(\vec{x})] &= (-1)\vec{x} \oplus (0) & [\mathrm{s}(\vec{x})] &= (2)\vec{x} \oplus (3) \end{aligned}$$

This interpretation is absolutely positive, gives us a decrease for the dependency pair

$$\begin{aligned} [\mathrm{cond}^\sharp(\mathrm{true}, x, y)] &= (\ \ 0)\vec{x} \oplus (-\infty)\vec{y} \oplus (2) \\ [\mathrm{cond}^\sharp(\mathrm{gr}(x, y), \mathrm{p}(x), \mathrm{s}(y))] &= (-1)\vec{x} \oplus (-\infty)\vec{y} \oplus (0) \end{aligned}$$

and all the original rules are oriented weakly. ◁

Remark 8.5. We discuss a variant which looks more liberal, but turns out to be equivalent to the one given here. We cannot allow $\mathbb{Z} \times \mathbb{A}_{\mathbb{Z}}^{d-1}$ for the domain, because it is not well-founded for $\gg$. So we can restrict the admissible range of negative values by some bound $c > -\infty$, and use the domain $\mathbb{A}_{\mathbb{Z}_{\geq c}} \times \mathbb{A}_{\mathbb{Z}}^{d-1}$ where $\mathbb{A}_{\mathbb{Z}_{\geq c}} := \{b \in \mathbb{A}_{\mathbb{Z}} \mid b \geq c\}$. Now to ensure that we stay within this domain we would demand that the first position of the constant vector of every interpretation is greater or equal than c.

Note however that this c can be fixed to 0 without any loss of generality as every interpretation using lower values in those positions can be "shifted" upwards. For any interpretation $[\cdot]$ and arctic number d construct an interpretation $[\cdot]'$ by $[t]' := [t] \otimes d$. This is obtained by going from $[f] = M_1\vec{x}_1 \oplus \ldots M_k\vec{x}_k \oplus \vec{c}$ to $[f]' = M_1\vec{x}_1 \oplus \ldots M_k\vec{x}_k \oplus \vec{c} \otimes d$. (A linear function with absolute part can be scaled by scaling the absolute part.) □

9 Quasi-Periodic Interpretations

Example 9.1. We consider the string rewriting system $S = \{bab \to a^3,\ a^3 \to b^3\}$, `Waldmann/jw1.srs` from TPDB, as a (running) example. Termination could not be established automatically by any of the programs taking part in the competition 2006. Then, Aleksey Nogin and Carl Witty produced a handwritten proof, that had been streamlined by Hans Zantema, and it had later been generalized into the method of *quasi-periodic interpretations* [39]. ◁

We recall the basic notion:

Definition 9.2. A function $f : \mathbb{N} \to \mathbb{N}$ is called *quasi-periodic* of *slope* s and *period* p if for all x, we have $f(x + p) = f(x) + sp$. ◇

In [39] it had been shown that quasi-periodic interpretations can prove termination of some rewrite systems for which no other proof was known (at the time). We now relate this approach to arctic matrix interpretations, by showing that they can simulate quasi-periodic interpretations of slope one for unary signatures.

Example 9.3. The dependency pairs transformation reduces the termination problem for S from Example 9.1 to the top termination problem $\mathrm{SN}(R_{\mathrm{top}}/S)$, with

$$R = \{Bab \to Aaa,\ Aaa \to Bbb\}$$

where all length-decreasing dependency pairs have already been removed. The proof given in [39] uses these quasi-periodic functions of period 3:

x	0	1	2	3	4	5	...
$[a](x) = [A](x)$	1	2	3	4	5	6	...
$[b](x) = [B](x)$	0	3	3	3	6	6	...

which induce these interpretations of the words in the rules:

x	0	1	2	3	4	5	...
$[Bab](x) = [bab](x)$	3	6	6	6	9	9	...
$[Aaa](x) = [aaa](x)$	3	4	5	6	7	8	...
$[Bbb](x) = [bbb](x)$	0	3	3	3	6	6	...

We infer that for all x, $[bab](x) \geq [aaa](x)$ and $[aaa](x) > [bbb](x)$, so there can not be infinitely many top applications of $Aaa \to Bbb$. This is the essential step in the termination proof. ◁

We give an encoding from weakly monotonic quasi-periodic functions of slope one to arctic matrices and show that it is a morphism (it maps composition to multiplication) and that it respects weak and strong compatibility with a string rewriting system.

9.1 Basic translation

Throughout, we fix the natural number $p > 0$ to be the period.

Then each $x \in \mathbb{N}$ has a unique representation $x = qp + r$ with $0 \leq r < p$.

We define a mapping

$$\begin{aligned} \mathrm{av} &: \mathbb{N} \to \mathbb{A}^p \\ \mathrm{av} &: x \mapsto (-\infty, \ldots, -\infty, \underbrace{q}_{\text{at position } r}, -\infty, \ldots, -\infty) \end{aligned}$$

In this section, vector indices start from 0 (not 1).

Example 9.4. For period $p = 3$, we have $\mathrm{av}(0) = (0, -\infty, -\infty)$ and $\mathrm{av}(4) = (-\infty, 1, -\infty)$. ◁

For a quasi-periodic function f we define its *associated arctic matrix* $[f]$ (of size $p \times p$) by giving its column vectors:

$$[f] \quad = \quad \left(\begin{array}{ccc} \mathrm{av}(f(0))^T & \ldots & \mathrm{av}(f(p-1))^T \end{array} \right)$$

Example 9.5. For period $p = 3$, consider the quasi-periodic functions

x	0	1	2	3	4	5	...
$f(x)$	1	2	4	4	5	7	...
$g(x)$	3	3	5	6	6	8	...

with associated matrices

$$[f] = \begin{pmatrix} -\infty & -\infty & -\infty \\ 0 & -\infty & 1 \\ -\infty & 0 & -\infty \end{pmatrix} \qquad [g] = \begin{pmatrix} 1 & 1 & -\infty \\ -\infty & -\infty & -\infty \\ -\infty & -\infty & 1 \end{pmatrix}$$ ◁

Lemma 9.6. If f is quasi-periodic of period p and slope one, then $[f] \otimes \mathrm{av}(x)^T = \mathrm{av}(f(x))^T$.

Proof. Let $x = pq + r$ with $0 \leq r < p$. Since the slope of f is one, we have $f(x) = pq + f(r)$ and we put $f(r) = pq' + r'$ with $0 \leq r' < p$.

We compute the entry at position i in $[f] \otimes \mathrm{av}(x)^T$. Since $\mathrm{av}(x)^T$ has exactly one finite entry, namely q at position r, we get q times the i-th position of the r-th column of $[f]$, which is $q \otimes \mathrm{av}(f(r))[i]$. This is finite exactly for $i = r'$, and then the value is $q \otimes q'$. So, the result vector is $\mathrm{av}(p(q+q') + r')$, and by the above, indeed $f(x) = pq + pq' + r'$. □

The mapping $[\cdot]$ is in fact a homomorphism:

Lemma 9.7. If f and g are both quasi-periodic functions of common period p and slope one, then $[f \circ g] = [g] \otimes [f]$.

Here, function composition is $(f \circ g) : x \mapsto g(f(x))$ and $\otimes$ is the (arctic) matrix product.

Proof. We compute row i of $[g] \otimes [f]$, which is $[g]$ times row i of $[f]$, being $[g] \otimes \mathrm{av}(f(i))^T$. By Lemma 9.6, this is $\mathrm{av}(g(f(i)))^T$. □

We remark that a matrix interpretation of this shape corresponds to a *complete and deterministic* weighted (word) automaton. This means that for each state and letter, there is exactly one transition with nonzero weight.

9.2 Weak Compatibility

Now we treat compatibility. Referring to Example 9.5, the function g is greater than the function f, but their associated matrices are not comparable w.r.t. $\gg$ or $\geq$. This will be repaired as follows. We start with weak compatibility.

For ease of presentation, we use arctic values below zero. We will see later that this can be removed.

We define an arctic triangular matrix of size $p \times p$ by

$$D \quad = \quad (\text{if } i \leq j \text{ then } 0 \text{ else } -1)_{i,j}$$

Example 9.8. For $p = 3$ we get:

$$D = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ -1 & -1 & 0 \end{pmatrix}$$ ◁

Lemma 9.9. For $x = pq + r$ with $0 \leq r < p$, we have

$$D \otimes \mathrm{av}(x)^T = (\underbrace{q, \ldots, q}_{r+1 \text{ entries}}, \underbrace{q-1, \ldots, q-1}_{p-r-1 \text{ entries}})^T$$

Proof. Entry number i (counting starts at 0) of the result vector is: if $i \leq r$ then q else $q - 1$. □

Lemma 9.10. If $x \leq y$, then $D \otimes \mathrm{av}(x)^T \leq D \otimes \mathrm{av}(y)^T$ component-wise.

Proof. Follows directly from Lemma 9.9. □

The matrices that arise here have a special shape:

Definition 9.11. An arctic matrix is called *flat* if each row $(x_1, \ldots, x_n)$ fulfills $x_1 \leq \ldots \leq x_n \leq x_1 + 1$ and each column $(y_1, \ldots, y_m)^T$ fulfills $y_m + 1 \geq y_1 \geq \ldots \geq y_m$. ◇

Note that we define this for any rectangular shape.

Lemma 9.12. If f is a weakly monotone quasi-periodic function of slope one, then $D \otimes [f]$ is flat.

Proof. By Lemma 9.9, each column of $D \otimes [f]$ has the required shape. For the shape of the rows, we argue as follows. The j-th and the $(j+1)$-th column of $D \otimes [f]$ are $D \otimes \mathrm{av}(f(j))^T$ and $D \otimes \mathrm{av}(f(j+1))^T$, respectively. By weak monotonicity of f, we have $f(j) \leq f(j+1)$, so by Lemma 9.10, each row of $D \otimes [f]$ is weakly increasing. Since f is monotonic, and also quasi-periodic of period p and slope one, we have $f(p-1) \leq f(p) = p + f(0)$. By Lemma 9.10,

$$D \otimes \mathrm{av}(f(p-1))^T \leq D \otimes \mathrm{av}(p + f(0))^T$$

Note that $\mathrm{av}(p + x) = \mathrm{av}(x) \otimes 1$, since arctic multiplication by 1 means just to increase each (finite) entry by one, therefore

$$D \otimes \mathrm{av}(f(p-1))^T \leq D \otimes \mathrm{av}(f(0))^T \otimes 1$$

so for each row index i, we have $(D \otimes [f])[i, p-1] \leq (D \otimes [f])[i, 0] + 1$. □

Lemma 9.13. If M is flat, then $M \otimes D = M$.

Proof. The entry at position (i, j) in $M \otimes D$ is the dot product of row i in M and column j in D. Let row i of M be $\vec{x} = (x_1, \ldots, x_p)$, Column j in D has shape

$$(\underbrace{0, \ldots, 0}_{j \text{ times}}, \underbrace{-1, \ldots, -1}_{p-j \text{ times}})^T$$

The dot product of these vectors is

$$\max\{x_1, \ldots, x_j, x_{j+1} - 1, \ldots, x_p - 1\}$$

By flatness of M, we have $x_1 \leq \ldots x_p \leq x_1 + 1$, so the maximum is realized by x_j. This is exactly the value of the entry at position (i, j) in M. □

Lemma 9.14. For any weakly monotonic quasi-periodic function f of slope one, $D \otimes [f] \otimes D = D \otimes [f]$.

Proof. By Lemma 9.12, $D \otimes [f]$ is flat. By Lemma 9.13, the claim follows. □

Definition 9.15. $M_D := D \otimes M$ ◇

Example 9.16. For f, g from Example 9.5,

$$[f]_D = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix} \qquad [g]_D = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$ ◁

Now we present two important properties of the translation $f \mapsto [f]_D$. It is a homomorphism (function composition corresponds to matrix multiplication), Lemma 9.17, and it respects the weak ordering, Lemma 9.18.

Lemma 9.17. $[f \circ g]_D = [g]_D \otimes [f]_D$

Proof. By Lemma 9.7, $[f \circ g]_D = ([g] \otimes [f])_D$. Denote $[f]$ by F and $[g]$ by G. Then $G_D F_D = DGDF = DGF = (GF)_D$ by Lemma 9.14. □

Lemma 9.18. If quasi-periodic functions f, g of period p and slope one fulfill $\forall x : f(x) \geq g(x)$, then $[f]_D \geq [g]_D$.

Proof. We consider column r. It has value $D \otimes \mathrm{av}(f(r))^T$ resp. $D \otimes \mathrm{av}(g(r))^T$, with $f(r) \geq g(r)$ by assumption. The result then follows from Lemma 9.10. □

The given translation $f \mapsto D \otimes [f]$ may create arctic matrices with negative entries. It can be verified that -1 is the *only* negative value that may ever appear, and that it is safe to replace it by $-\infty$, in order to obtain an interpretation with arctic naturals.

9.3 Strict Compatibility

Again referring to Example 9.5, the function g is strictly greater than the function f, but as Example 9.16 shows, $[g]_D \not\gg [f]_D$. By modifying the interpretation of some symbols, we obtain strict compatibility. For easier presentation, we use rational weights. Weights can be made integral by scaling: this is multiplication by a constant, resp. arctic exponentiation.

Define E as the $p \times p$ square matrix where all rows are equal to vector $F = (0, 1/p, \ldots, (p-1)/p)$. The interesting property of F is:

Lemma 9.19. $F \otimes \mathrm{av}(x)^T = x/p$.

Proof. Let $x = pq + r$. The vector $\mathrm{av}(x)$ has only one non-zero entry, namely q at position r. Then $F \otimes \mathrm{av}(x)^T = r/p + q = x/p$. □

This implies

Lemma 9.20. If $x, y \in \mathbb{N}$ and $x > y$, then $F \otimes \mathrm{av}(x)^T > F \otimes \mathrm{av}(y)^T$ and $E \otimes \mathrm{av}(x)^T \gg E \otimes \mathrm{av}(y)^T$.

Proof. The first statement follows from the previous lemma. Then the second statement follows, as all rows of E are equal to F. □

Now the application is that we can multiply an interpretation (that was translated according to $f \mapsto [f]_D$) from the left by E, to get the desired relation:

Lemma 9.21. If quasi-periodic functions f, g of period p and slope one fulfill $f < g$ point-wise, then $E \otimes [f]_D \ll E \otimes [g]_D$.

Proof. We have $E \otimes [f]_D = ED[f] = E[f]$, since E is flat and Lemma 9.13 applies. Column r of $E[f]$ is the product of E and column r of $[f]$, thus $E \otimes \mathrm{av}(f(r))^T$. This is to be compared with $E \otimes \mathrm{av}(g(r))^T$, so we apply Lemma 9.20. □

9.4 Putting it all together

While we achieve weak compatibility (w.r.t. $\geq$) by the translation $[\cdot]_D$, we get strict compatibility (w.r.t. $\gg$) only for the shape of top rewrite relations that arise from the dependency pair transformation.

Theorem 9.22. Given a weakly monotonic quasi-periodic interpretation of period p and slope one that is weakly compatible with S and R and strictly compatible with R', where the top symbols of $R \cup R'$ do not occur in S, there is an arctic matrix interpretation of dimension p that fulfills the conditions of Theorem 7.1: it is weakly compatible with S and R, and strictly compatible with R'.

Proof. This interpretation is obtained by taking as the translation of a non-top symbol interpretation f the matrix $[f]_D$, and for a top symbol, the matrix $E \otimes [f]_D$. This interpretation is somewhere finite since the top left entry of each matrix is finite. This follows from $f(0) \geq 0$. This translation computes the correct values by Lemma 9.17, and we get weak compatibility by Lemma 9.18 as well as strict compatibility by Lemma 9.21. □

Example 9.23. For the quasi-periodic interpretation from Example 9.3,

$$[a] = \begin{pmatrix} -\infty & -\infty & 1 \\ 0 & -\infty & -\infty \\ -\infty & 0 & -\infty \end{pmatrix} \qquad [a]_D = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$[b] = \begin{pmatrix} 0 & 1 & 1 \\ -\infty & -\infty & -\infty \\ -\infty & -\infty & -\infty \end{pmatrix} \qquad [b]_D = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$[a^3]_D = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad [b^3]_D = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$E \otimes [a^3]_D = \begin{pmatrix} 1 & 4/3 & 5/3 \\ 1 & 4/3 & 5/3 \\ 1 & 4/3 & 5/3 \end{pmatrix} \qquad E \otimes [b^3]_D = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$E \otimes [a]_D = \begin{pmatrix} 1/3 & 2/3 & 1 \\ 1/3 & 2/3 & 1 \\ 1/3 & 2/3 & 1 \end{pmatrix} \qquad E \otimes [b]_D = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

◁

Remark 9.24. We comment on the effect of restricting the slope to one. Quasi-periodic interpretations of higher slope cannot be represented by arctic matrix interpretations: for instance, the function $f : x \mapsto 2x$ is a quasi-periodic function of slope 2 and period 1 (trivially), and iterated application of f gives exponentially increasing values, while arctic matrices (of any dimension) can only give linear growth. On the other hand, as has been remarked already in [39], if quasi-periodic functions are applied together with other termination methods, restricting the slope to one does not seem to reduce the power of the method too much. There are several hard termination problems where slope one is sufficient. □

10 Certification

The theory developed in this paper for proving termination with arctic (below-zero) interpretations (*i.e.*, Theorem 7.1 and Theorem 8.3) is accompanied by formal proofs in the Coq proof assistant [34, 6]. Coq is a proof assistant/checker based on the Calculus of Inductive Constructions (CIC) [33] — a very expressive logic supporting simple, inductive, dependent and polymorphic types.

The certification has been carried out within the CoLoR project [7]. The main part of CoLoR is a library of termination techniques formalized in Coq. The aim of the project is to gather such formalizations within this library and then use them to certify concrete proofs produced by some existing termination provers.

This is accomplished by introducing an intermediate format for termination proofs. Automated termination provers only need to support this format (which should be easy to accomplish) and then Rainbow, a simple tool developed within the CoLoR project, transforms proofs in this format to actual Coq scripts certifying termination. Then Coq is run as a proof checker to verify correctness of the proof reported by the termination tool, with the use of the termination criteria formalized in the CoLoR library.

We formalized the arctic techniques for proving termination of term rewriting presented in this paper, so the criteria corresponding to Theorems 7.1 and 8.3 but not that of Theorem 6.7.

Let us illustrate the CoLoR approach to certification on the termination criteria developed in this paper. An arctic termination proof requires providing an arctic interpretation, Definition 4.4, which consists of:

- a natural number d, indicating the dimension of vectors and matrices that are used,

- a theorem to be used: either Theorem 7.1 or Theorem 8.3 and
- for every symbol f the arctic function corresponding to it (over $\mathbb{A}_{\mathbb{N}}$ in case of Theorem 7.1 or over $\mathbb{A}_{\mathbb{Z}}$ for Theorem 8.3), *i.e.*, the constant vector $\vec{c}$ and its linear factors $M_1, \ldots, M_n$, where n is the arity of f.

Such arctic interpretation can be specified by means of a formal grammar, which is part of the aforementioned proof format. Then proofs like those presented in Example 7.2 or Example 8.4 can be easily expressed in this format and translated by Rainbow to formal Coq proofs, allowing to check their correctness with Coq. Similar transformation can be applied to termination proofs generated by termination provers, allowing a fully automatic certification of their results. We implemented this in the termination prover Matchbox [36] and we report on the results in the following section.

The basis of this formalization work was the certification of the matrix interpretations method [27], the method introduced shortly in Example 3.6, which consists of formalizations of:

- a semiring structure,
- vectors and matrices over arbitrary semirings of coefficients,
- the monotone algebras framework and
- the matrix interpretation method.

The framework of monotone algebras was used without any changes at all. Vectors and matrices were formalized in [27] for arbitrary semirings, however all the results involving orders were developed for the usual orders on natural numbers, as used in the matrix interpretation method. So the first step in the formalization process was to generalize the semiring structure to a semiring equipped with two orders $(>, \geq)$ and to adequately generalize results on vectors and matrices. Then the arctic semiring was developed in this setting.

As for the technique itself it has a lot in common with the technique of matrix interpretations. Therefore the common parts were extracted to a module `MatrixBasedInt` which was then specialized to the matrix interpretation method (`MatrixInt`) and to a basis for arctic based methods (`ArcticBasedInt`), which was narrowed down to the methods of arctic interpretations (`ArcticInt`) and arctic below-zero interpretations (`ArcticBZInt`). This hierarchy is depicted in Figure 1.

We did not yet formalize the developments of Section 9 on quasi-periodic interpretations. Note that this is not strictly necessary: one can still certify quasi-periodic termination proofs with CoLoR indirectly: just carry out the transformation according to Theorem 9.22, and submit the corresponding arctic interpretation for certification. This has been realized by the termination prover Matchbox in the Certified Termination Competition for string rewriting in 2008.

Considering the extension of the formal proof format and the Rainbow tool it was minimal. The format for the matrix interpretation proofs was already developed in [27] and it essentially requires to provide matrix interpretations for all the

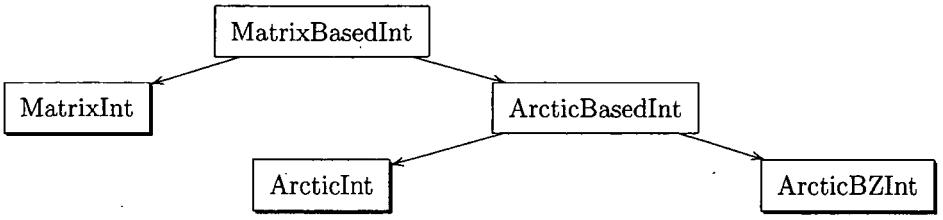


Figure 1: Hierarchy of different matrix-based methods in CoLoR.

function symbols in the signature. The format for arctic interpretations is the same except that:

- it indicates which matrix-based method is to be used, indicated by different XML tags (as the common proof format of CoLoR is in the end specified using XML syntax),
- the entries of vectors and matrices are from a different domain.

Below we illustrate the performance of the winning tools in the certified competitions of 2007 and 2008. In case of the 2008 competition we narrow down the results only to the set of problems used in 2007, to make the figures and tools' performances comparable.

category	year	winner	termination proofs
TRS	2007	CoLoR + TPA	$354/975 \approx 36.3\%$
	2008	CoLoR + AProVE	$420/975 \approx 43.1\%$
SRS	2007	Matchbox (uncertified)	$337/517 \approx 65.2\%$
	2008	CoLoR + Matchbox	$354/517 \approx 68.5\%$

Figure 2: Performance of the winners of the certified competition in 2007 and 2008.

In the SRS category the only improvement in Matchbox [36] from 2007 to 2008 was the addition of arctic interpretations, and the transformation from quasi-periodic to arctic interpretations.

Even more importantly we are comparing here the un-certified Matchbox 2007 (there was no certified SRS category in 2007) with its certified 2008 counterpart. That means that some features not supported by CoLoR had to be switched off in Matchbox for the 2008 competition and still it performed better than in the previous year. Roughly half of the proofs produced by Matchbox involved arctic interpretations, showing the importance of this technique for string rewriting.

The interpretation of the results in the TRS category is much more difficult. TPA [24] and AProVE [16] are very different tools using different proof search strategies. Also apart from the addition of arctic interpretations, AProVE could take advantage of the dependency graph decomposition technique that was not available in CoLoR in the previous year.

In the following section we will present the experimental data comparing performance of Matchbox with different sets of techniques in use, which will allow for a better evaluation of the impact of arctic interpretations on termination proving power.

11 Implementation

The implementation in Matchbox follows the scheme described in [13]. The constraint problem for the arctic interpretation is translated to a constraint problem for matrices, for arctic numbers and, finally, for Boolean variables. This is then solved by Minisat [11].

An arctic number is represented by a pair $a = (b; v_0, v_1, \ldots, v_n)$ where b is a Boolean value and $v_0, \ldots, v_n$ is a sequence of Booleans (all numbers have fixed bit-width). If b is 1, then a represents $-\infty$, if b is 0, then a represents the binary value of $v_0, \ldots, v_n$.

To represent integers, we use two's complement representation, *i.e.*, the most significant bit is the "sign bit".

Note that implementation of max/plus operation is less expensive than standard plus/times: with a binary representation both max and plus can be computed (encoded) with a linear size formula (whereas a naive implementation of the standard multiplication requires quadratic size and asymptotically better schemes do not pay off for small bit widths).

It is useful to require the following, for each arctic number $a = (b, v)$: if the infinity bit b is set, then $v = 0$. Then $(b, v) \oplus (b', v') = (b \wedge b', \max(v, v'))$. For $(b, v) \otimes (b', v')$ we compute $c = b \vee b'$, $u = (u_0, \ldots, u_n) = v + v'$ and the result is $(c; \neg c \wedge u_0, \ldots, \neg c \wedge u_n)$.

To represent arctic integers, we use a similar convention: if the infinity flag b is set, we require that the number v represents the lowest value of its range.

The following table lists the numbers of certified proofs that we obtain with the dependency pair method (without analyzing the dependency graph) and the following matrix methods: (s)tandard, (a)rctic, below (z)ero. The problem set used for this experiment is the TPDB of the 2007 competition, so the problems correspond to those presented in Figure 2.

Runs were executed on a single core of an Intel X5365 processor running at 3GHz. All proofs are available for inspection at the Matchbox web page [36]. In all cases we used standard matrices of dimension 1 and 2 to remove rules before the dependency pair transformation, and then matrix dimensions d from 1 up; with numbers of bit width 3, and a timeout of $1+d^2$ seconds for each individual attempt.

Here, we count only verified proofs, so we are missing about 3 to 5 proofs where Coq does not finish in reasonable time.

For a SRS $\mathcal{R}$ we consider $\mathrm{reverse}(\mathcal{R}) = \{\mathrm{reverse}(l) \to \mathrm{reverse}(r) \mid (l \to r) \in \mathcal{R}\}$. Clearly this transformation preserves termination both ways (the implication $\mathrm{SN}(\mathrm{reverse}(\mathcal{R})) \implies \mathrm{SN}(\mathcal{R})$ has been formalized in CoLoR). Half of the allotted

problem set	time	number of proofs found by the method			
		s	sa	sz	saz
975 TRS	1 min	361	376	388	389
	10 min	365	381	393	394
517 SRS	1 min	178	312	298	320
	10 min	185	349	323	354

Figure 3: Performance of Matchbox using different versions of matrix (arctic) interpretations.

time is spent for each of $\mathcal{R}$ and reverse($\mathcal{R}$). This increases the score considerably (by about one third).

In the previous section we already mentioned that the method of arctic interpretations was implemented in AProVE. Recently it was also incorporated into T$_{\text{T}}$T$_2$ and both those provers used it in the termination competition of 2008, where they took the first two places in the main categories (full termination of term/string rewriting systems).

12 Discussion

Arctic naturals form a sub-semiring of arctic integers. So the question comes up whether Theorem 8.3 subsumes Theorem 7.1. Note that the prerequisites for both theorems are incomparable. Still there might be a method to construct from a somewhere-finite interpretation (above zero) an equivalent absolutely positive interpretation (below zero). We are not aware of any. Experience with implementation shows that it is useful to have both methods, especially for string rewriting. Naturals are easier to handle than integers because they do not require signed arithmetics. So typically we can increase the bit width or the matrix dimension for naturals. Our implementation finds several proofs according to Theorem 7.1 where it fails to find a proof according to Theorem 8.3 and vice-versa.

It is interesting to ask whether the preconditions of Theorems 6.7,7.1,8.3 can be weakened. We discussed one variant in Remark 8.5. In general, a linear interpretation $[\cdot]$ with coefficients in $\mathbb{A}_{\mathbb{N}}$ ($\mathbb{A}_{\mathbb{Z}}$ respectively) is admissible for a termination proof if for each ground term t, the value $[t]$ is finite (positive, respectively). This is in fact a reachability problem for weighted (tree) automata. It is decidable for interpretations on arctic naturals, but it is undecidable for arctic integers (this follows from a result of Krob [29] on tropical word automata). In our setting, we do not guess an interpretation and then decide whether it is admissible. Rather, we have to formulate the decision algorithm as part of the constraint system for the interpretation. Therefore we chose sharper conditions on interpretations that imply finiteness (positiveness, respectively) and have an easy constraint encoding.

Another question is the relation of the standard matrix method to the arctic matrix method(s). Performance of our implementation suggests that neither method subsumes the other, but this may well be a problem of computing resources, as we hardly reach matrix dimension 5 and bit width 3.

As for the relation to other termination methods (*e.g.*, path orderings), the only information we have is that arctic (and other) matrix methods can do non-simple termination, while path orders and polynomial interpretations cannot; and on the other hand, the arctic matrix method implies a linear bound on derivational complexity (see below), which is easily surpassed by path orders and other interpretations.

The full arctic termination method bounds lengths of derivations:

Lemma 12.1. For a rewriting system $\mathcal{R}$ that fulfills the requirements of Theorem 6.7 for $\mathcal{S} = \emptyset$, the derivational complexity of $\mathcal{R}$ is linear.

Proof. For a finite arctic vector $\vec{x} = (x_1, \ldots, x_k)$, define $|\vec{x}| = \max(x_1, \ldots, x_k)$.

Then $|\vec{x} \oplus \vec{y}| \leq \max(|\vec{x}|, |\vec{y}|)$ and $|\vec{x} \otimes \vec{y}^T| \leq |\vec{x}| + |\vec{y}|$.

For a finite arctic matrix A of dimension $k \times k$, define $|A| = \max\{A[i,j] \mid 1 \leq i,j \leq k\}$. Then $|A \otimes \vec{x}| \leq |A| + |\vec{x}|$ and $|A \otimes B| \leq |A| + |B|$.

For an interpretation $[\cdot]$ of some signature Σ, and any word $w \in \Sigma^*$, this implies that $|[w]| \leq c \cdot |w|$ where $c = \max\{|[f]| : f \in \Sigma\}$.

Now we remark that $u \to_{\mathcal{R}} v$ implies $[u] \gg [v]$, and $\vec{x} \gg \vec{y}$ implies $|\vec{x}| > |\vec{y}|$. Thus the derivational complexity of $\mathcal{R}$ is linear: any derivation starting from u has at most $c \cdot |u|$ steps. □

This means that rewriting systems with higher derivational complexity (*e.g.*, quadratic: $\{\mathrm{a\,b} \to \mathrm{b\,a}\}$, or exponential $\{\mathrm{a\,b} \to \mathrm{b}^2\,\mathrm{a}\}$) do not admit an arctic termination proof. Note that both these systems admit a standard matrix proof.

It seems very difficult to combine this argument with the dependency pair method, as it can drastically alter (*i.e.*, reduce) derivational complexity [32].

Example 12.2. The following rewriting system [21] has a derivational complexity that is not primitive recursive:

$$\{\mathrm{s}(x) + (y + z) \to x + (\mathrm{s}(\mathrm{s}(y)) + z), \mathrm{s}(x) + (y + (z + w)) \to x + (z + (y + w))\}$$

and still it has, after dependency pairs transformation, an easy termination proof by "counting symbols" [13]. Note however that arctic interpretations cannot count globally: to compute the interpretation $[f(t_1, t_2)]$, it is impossible to add values from subtrees $[t_1], [t_2]$, as we can only take the maximum of $[t_1], [t_2]$. Yet we find an arctic proof, as follows. The given system is in fact an encoding of a length-preserving string rewriting system on the infinite alphabet $\mathbb{N}$. Both rules keep the right spine of terms (corresponding to the length of the simulated string) intact, so we can remove dependency pairs that shrink it, using the interpretation $[+](x, y) = y \otimes 1$. We are left with two dependency pairs (that directly correspond to the original rules). They can be handled by $[+](x, y) = x$ and $[\mathrm{s}](x) = x \otimes 1$. So instead of numbers of symbols, we were just using path lengths. ◁

Max/plus polynomials have been used by Amadio [3] as quasi-interpretations (*i.e.*, functions are weakly monotone), to bound the space complexity of derivations. Proving termination directly was not intended.

13 Conclusions

We presented the arctic interpretations method for proving termination of term rewriting. It is based on the matrix interpretation method [13] where the usual plus/times operations on $\mathbb{N}$ are generalized to an arbitrary semiring, in this case instantiated by the arctic semiring (max/plus algebra) on $\{-\infty\} \cup \mathbb{N}$.

Matrix interpretations are an efficient realization of a certain class of weighted tree automata. It remains a subject of further study to characterize the family of weighted tree languages that can be represented in that way.

We also generalized this to arctic integers. This generalization allowed us to solve 10 of Beerendonk/* examples that are difficult to prove terminating and thus far could only be solved by AProVE with the Bounded Increase [17] technique, dedicated to such class of problems coming from transformations from imperative programs and with polynomial interpretations with rational coefficients [30].

Our presentation of the theory is accompanied by a formalization in the Coq proof assistant. By becoming part of the CoLoR project this formalization allows us to formally verify termination proofs involving the arctic matrix method. It was evaluated in the certified category of the termination competition in 2008 and turned out to be a crucial contribution allowing CoLoR to win with the competing certification back-end, A3PAT [8].

We want to remark here that all performance data and all examples presented in this paper were collected from problems of TPDB and we did not "cook up" any special examples to show off the arctic method. The emphasis of these examples (in fact, of the whole paper) is not to provide termination proofs where none were known before, but rather to provide certified (and often conceptually simpler) termination proofs where only uncertified proofs were available up to now.

References

[1] The termination competition. `http://www.termination-portal.org/wiki/Termination_Competition`.

[2] Termination problems data base. `http://dev.aspsimon.org/projects/termcomp/downloads/`.

[3] Amadio, R. M. Synthesis of max-plus quasi-interpretations. *Fundamenta Informaticae*, 65(1-2):29–60, 2005.

[4] Arts, T. and Giesl, J. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.

[5] Baader, F. and Nipkow, T. *Term Rewriting and All That.* Cambridge University Press, 1998.

[6] Bertot, Y. and Castéran, P. *Coq'Art: The Calculus of Inductive Constructions.* EATCS Texts in Theoretical Computer Science. Springer, 2004.

[7] Blanqui, F., Delobel, W., Coupet-Grimal, S., Hinderer, S., and Koprowski, A. CoLoR, a Coq library on rewriting and termination. In *Workshop on Termination (WST)*, 2006. `http://color.loria.fr`.

[8] Contejean, É., Courtieu, P., Forest, J., Pons, O., and Urbain, X. Certification of automated termination proofs. In *International Symposium on Frontiers of Combining Systems (FroCoS)*, volume 4720 of *Lecture Notes in Computer Science*, pages 148–162, 2007.

[9] Droste, M., Pech, C., and Vogler, H. A Kleene theorem for weighted tree automata. *Theory of Computing Systems*, 38(1):1–38, 2005.

[10] Droste, M. and Vogler, H. Weighted tree automata and weighted logics. *Theoretical Computer Science*, 366(3):228–247, 2006.

[11] Eén, N. and Sörensson, N. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing (SAT)*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518, 2003.

[12] Endrullis, J. Jambox. `http://joerg.endrullis.de`.

[13] Endrullis, J., Waldmann, J., and Zantema, H. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(2–3):195–220, 2008.

[14] Fuchs, L. *Partially Ordered Algebraic Systems.* Addison-Wesley, 1962.

[15] Gaubert, S. and Plus, M. Methods and applications of (max, +) linear algebra. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1200 of *Lecture Notes in Computer Science*, pages 261–282, 1997.

[16] Giesl, J., Thiemann, R., Schneider-Kamp, P., and Falke, S. Automated termination proofs with AProVE. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 3091 of *Lecture Notes in Computer Science*, pages 210–220, 2004.

[17] Giesl, J., Thiemann, R., Swiderski, S., and Schneider-Kamp, P. Proving termination by bounded increase. In *International Conference on Automated Deduction (CADE)*, volume 4603 of *Lecture Notes in Computer Science*, pages 443–459, 2007.

[18] Golan, J. S. *Semirings and their Applications.* Kluwer, 1999.

[19] Hirokawa, N. and Middeldorp, A. Polynomial interpretations with negative coefficients. In *International Conference on Artificial Intelligence and Symbolic Computation (AISC)*, volume 3249 of *Lecture Notes in Computer Science*, pages 185–198, 2004.

[20] Hirokawa, N. and Middeldorp, A. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.

[21] Hofbauer, D. and Lautemann, C. Termination proofs and the length of derivations. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 355 of *Lecture Notes in Computer Science*, pages 167–177, 1989.

[22] Hofbauer, D. and Waldmann, J. Termination of string rewriting with matrix interpretations. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 4098 of *Lecture Notes in Computer Science*, pages 328–342, 2006.

[23] Hong, H. and Jakuš, D. Testing positiveness of polynomials. *Journal of Automated Reasoning*, 21(1):23–38, 1998.

[24] Koprowski, A. TPA: Termination proved automatically. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 4098 of *Lecture Notes in Computer Science*, pages 257–266, 2006. `http://www.win.tue.nl/tpa`.

[25] Koprowski, A. and Waldmann, J. Arctic termination ... below zero. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 5117 of *Lecture Notes in Computer Science*, pages 202–216, 2008.

[26] Koprowski, A. and Waldmann, J. Max/plus tree automata for termination of term rewriting. In *Workshop on Weighted Automata Theory and Applications (WATA)*, 2008.

[27] Koprowski, A. and Zantema, H. Certification of proving termination of term rewriting by matrix interpretations. In *Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 4910 of *Lecture Notes in Computer Science*, pages 328–339, 2008.

[28] Korp, M., Sternagel, C., Zankl, H., and Middeldorp, A. Tyrolean Termination Tool 2. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304, 2009. See `http://colo6-c703.uibk.ac.at/ttt2`.

[29] Krob, D. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 623 of *Lecture Notes in Computer Science*, pages 101–112, 1992.

[30] Lucas, S. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO: Theoretical Informatics and Applications*, 39(3):547–586, 2005.

[31] Marché, C., Waldmann, J., and Zantema, H. The termination competition 2007. `http://www.imn.htwk-leipzig.de/~waldmann/talk/07/wst/competition`.

[32] Moser, G. and Schnabl, A. The derivational complexity induced by the dependency pair method. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 5595 of *Lecture Notes in Computer Science*, pages 255–269, 2009.

[33] Paulin-Mohring, C. Inductive definitions in the system Coq - rules and properties. In *International Conference on Typed Lambda Calculi and Applications (TLCA)*, volume 664 of *Lecture Notes in Computer Science*, pages 328–345, 1993.

[34] The Coq Development Team. The Coq proof assistant: Reference manual, version 8.2. `http://coq.inria.fr`, 2009.

[35] Thiemann, R., Zantema, H., Giesl, J., and Schneider-Kamp, P. Adding constants to string rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 19(1):27–38, 2008.

[36] Waldmann, J. Matchbox: A tool for match-bounded string rewriting. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 3091 of *Lecture Notes in Computer Science*, pages 85–94, 2004. `http://dfa.imn.htwk-leipzig.de/matchbox`.

[37] Waldmann, J. Arctic termination. In *Workshop on Termination (WST)*, 2007.

[38] Waldmann, J. Weighted automata for proving termination of string rewriting. *Journal of Automata, Languages and Combinatorics*, 12(4):545–570, 2007.

[39] Zantema, H. and Waldmann, J. Termination by quasi-periodic interpretations. In *International Conference on Rewriting Techniques and Applications (RTA)*, volume 4533 of *Lecture Notes in Computer Science*, pages 404–418, 2007.

Received 1st August 2008

Acta Cybernetica 19 (2009) 393–410.

Weighted and Unweighted Trace Automata

Dietrich Kuske*

Abstract

We reprove Droste & Gastin's characterisation from [3] of the behaviors of weighted trace automata by certain rational expressions. This proof shows how to derive their result on *weighted* trace automata as a corollary to the *unweighted* counterpart shown by Ochmański.

Keywords: weighted automata, Mazurkiewicz traces

1 Introduction

A large body of theoretical computer science deals with properties of languages as sets of finite words. These words can be understood as the sequence of events performed by some system. This modelling works fine for sequential systems because of the linear nature of words. Mazurkiewicz [11] proposed a generalization of words nowadays called Mazurkiewicz traces that allows to also model some concurrency. Since its introduction, much work has been devoted to the transfer of results on word languages to trace languages (cf. [6]). One such result is Kleene's theorem [8] equating the recognizable and the rational languages. Ochmański [12] succeeded in transferring this result to trace languages showing that the recognizable trace languages are precisely the c-rational ones.

For sequential systems, it is not just interesting to ask whether a particular word is generated, but also to know the number of different ways it can be generated. This question developed into the theory of weighted automata and formal power series (cf. [14, 9, 1, 5]). A fundamental result is Schützenberger's theorem [13], equating the behaviors of weighted automata with the set of rational formal power series.

These two distinct generalizations of Kleene's theorem were re-joint by Droste & Gastin [3] who investigated weighted trace automata and formal power series over partially commuting variables.

The theorems by Kleene, by Schützenberger, by Ochmański, and by Droste & Gastin characterize the recognizable languages, formal power series, trace languages, or formal power series over partially commuting variables by certain rational operations. All the proofs follow the line of Kleene's proof (namely showing the

*Institut für Informatik, Universität Leipzig

closure of recognizable objects under the respective operations) albeit with non-trivial additions. An exception to this is the recent proof of the result by Droste & Gastin that Berstel & Reutenauer gave in [2]: they extend Brzozowski's derivations to also handle weights and partial commutation.

In this paper, we present another alternative proof of Droste & Gastin's characterisation of the behavior of weighted trace automata. The *novelty* lies in the fact that we derive their result as a corollary to Ochmański's theorem. In other words, we derive a result on *weighted* trace automata from a theorem on *unweighted* trace automata. This refines the methodology introduced in [10] where I similarly derived Schützenberger's theorem from Kleene's theorem.

The idea is as follows (see later sections for missing definitions): If $\mathcal{A}$ is a weighted trace automaton, then the set of paths from some initial to some final state forms a regular language L. We define conditions (T1-3) that formalize the relation between this language and the behavior of $\mathcal{A}$ (see Lemma 4.1). From a rational expression for L, we get an mc-rational expression for the behavior of $\mathcal{A}$ (see proof of Theorem 4.1). Conversely, let E be some c- or mc-expression. Then we construct a language L (Sections 3.2 and 3.3) satisfying (T1-3). The crucial point is that the language L gives rise to a weighted trace automaton whose behavior equals the semantics of E (Prop. 3.1).

2 Definitions

2.1 Weighted trace automata and their behavior

A structure $(S, +, \cdot, 0, 1)$ is a *semiring* if $(S, +, 0)$ is a commutative monoid, $(S, \cdot, 1)$ a monoid, $\cdot$ is both left- and right-distributive over $+$, and $0 \cdot k = k \cdot 0 = 0$ for all $k \in S$. If there is no ambiguity, we denote a semiring just by S. A semiring is *commutative* if $(S, \cdot, 1)$ is commutative; it is *idempotent* if $k + k = k$ for all $k \in S$.

An *independence alphabet* is a pair (Σ, I) where Σ is some alphabet and $I \subseteq \Sigma^2$ is an irreflexive and symmetric *independence relation*. Then $D = \Sigma^2 \setminus I$ is the complementary *dependence relation* which is reflexive and symmetric. Then $\sim$ denotes the least congruence relation on the free semigroup Σ^+ with $ab \sim ba$ for all $a, b \in \Sigma$ with $(a, b) \in I$. The quotient $\mathbb{M}^+(\Sigma, I) = \Sigma^+/\sim$ is the *trace semigroup generated by* (Σ, I)[1]; its elements are equivalence classes $[u]$ of words $u \in \Sigma^+$. Note that the semigroups $\mathbb{M}^+(\Sigma, \emptyset)$ and $(\Sigma^+, \cdot)$ are naturally isomorphic and we will identify the element $[u] = \{u\}$ of $\mathbb{M}^+(\Sigma, \emptyset)$ with the word $u \in \Sigma^+$. A language $L \subseteq \Sigma^+$ is *I-closed* if $u \sim v$ and $v \in L$ imply $u \in L$, i.e., $L = \bigcup_{v \in L} [v]$. Similarly, a function $\mu : \Sigma^+ \to X$ to some set X is *I-closed* if $u \sim v$ implies $\mu(u) = \mu(v)$. For $u \in \Sigma^+$, let $\mathrm{alph}(u)$ denote the alphabet of the word u, i.e., the set of letters occurring in u. Then $u \sim v$ implies $\mathrm{alph}(u) = \mathrm{alph}(v)$ which allows to set $\mathrm{alph}([u]) = \mathrm{alph}(u)$.

[1] Droste & Gastin work in the trace monoid $\mathbb{M}(\Sigma, I) = \mathbb{M}^+(\Sigma, I) \cup \{1\}$, but all their results hold with minor and obvious changes also in the trace semigroup (see Remark 2.1). We prefer to work in this trace semigroup since this eliminates the repetitive special handling of the unit element.

A *weighted trace automaton* over the independence alphabet (Σ, I) is a tuple $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ where Q is a finite and nonempty set of *states*, Σ is some alphabet, $\lambda \in S^{1\times Q}$ is a row vector, $\mu : \Sigma^+ \to (S^{Q\times Q}, \cdot)$ is an I-closed (semigroup-) homomorphism, and $\gamma \in S^{Q\times 1}$ is a column vector. Its (Σ, I)-*behavior* $\|\mathcal{A}\|_{(\Sigma,I)}$ is a mapping from $\mathbb{M}^+(\Sigma, I)$ into S given by $\|\mathcal{A}\|_{(\Sigma,I)}([u]) = \lambda \cdot \mu(u) \cdot \gamma$ for all $u \in \Sigma^+$ (since $\mu(u) = \mu(v)$ for $u \sim v$, this is well-defined). Note that every weighted trace automaton over (Σ, I) is also a weighted trace automaton over $(\Sigma, \emptyset)$. If (Σ, I) is clear from the context, the *trace behavior of* $\mathcal{A}$ is the function $\|\mathcal{A}\|_{\mathrm{T}} = \|\mathcal{A}\|_{(\Sigma,I)} : \mathbb{M}^+(\Sigma, I) \to S$ and the *word behavior of* $\mathcal{A}$ is the function $\|\mathcal{A}\|_{\mathrm{W}} = \|\mathcal{A}\|_{(\Sigma,\emptyset)} : \Sigma^+ \to S$. Then the trace and the word-behaviors are directly related by $\|\mathcal{A}\|_{\mathrm{T}}([u]) = \|\mathcal{A}\|_{\mathrm{W}}(u)$ for all $u \in \Sigma^+$.

For $p, q \in Q$ and $a \in \Sigma$, we say (p, a, q) is a *transition* of $\mathcal{A}$ if $\mu(a)_{p,q} \neq 0$. A *path of length* m is a sequence $U = (p_i, a_i, p_{i+1})_{1\leq i\leq m}$ of transitions; its label is the word $\pi(U) = a_1 a_2 \ldots a_m$ and its weight is $c(U) = \prod_{1\leq i<m} \mu(a_i)_{p_i,p_{i+1}}$. Then the trace behavior of $\mathcal{A}$ can also be described in terms of these paths, namely we have

$$\|\mathcal{A}\|_{\mathrm{T}}([u]) = \sum \left(\lambda(\iota) \cdot c(U) \cdot \gamma(f) \;\middle|\; \begin{array}{l} \iota, f \in Q, U \text{ is a path from} \\ \iota \text{ to } f \text{ with } \pi(U) = u \end{array} \right) \tag{1}$$

for any $u \in \Sigma^+$ ([7, Cor. VI.6.2]) since $\|\mathcal{A}\|_{\mathrm{T}}([u]) = \|\mathcal{A}\|_{\mathrm{W}}(u)$.

Mappings s from $\mathbb{M}^+(\Sigma, I)$ into a semiring S can be considered as formal power series in partially commuting variables (fps for short). In this context, one usually writes $(s, [u])$ for the value $s([u])$ and $S\langle\langle \mathbb{M}^+(\Sigma, I)\rangle\rangle$ for the set of all formal power series. For $s, t \in S\langle\langle \mathbb{M}^+(\Sigma, I)\rangle\rangle$ and $A \subseteq \Sigma$, we next define formal power series $s+t$, $s \cdot t$, s^+, and $(s)_A$. To this aim, let $x \in \mathbb{M}^+(\Sigma, I)$ and set

$$(s + t, x) = (s, x) + (t, x) \qquad (s \cdot t, x) = \sum_{\substack{y,z\in\mathbb{M}^+(\Sigma,I) \\ x=yz}} (s, y) \cdot (t, z)$$

$$((s)_A, x) = \begin{cases} (s, x) & \text{if alph}(x) = A \\ 0 & \text{otherwise} \end{cases} \qquad (s^+, x) = \sum_{1\leq i\leq |x|} (s^i, x)$$

where s^i denotes the i^{th} power of the formal power series s.

An *expression* is a term using the constants ka for $k \in S$ and $a \in \Sigma$, the binary operations $+$ and $\cdot$, and the unary operations $(\,)_A$ and $^+$. Any such expression E can be interpreted as a fps $[\![E]\!]_{(\Sigma,I)} \in S\langle\langle \mathbb{M}^+(\Sigma, I)\rangle\rangle$, the (Σ, I)-*semantics of* E. More formally, we defined inductively

$$([\![ka]\!]_{(\Sigma,I)}, x) = \begin{cases} k & \text{if } x = [a] \\ 0 & \text{otherwise} \end{cases} \qquad ([\![E + F]\!]_{(\Sigma,I)}, x) = ([\![E]\!]_{(\Sigma,I)}, x) + ([\![F]\!]_{(\Sigma,I)}, x)$$

$$([\![E^+]\!]_{(\Sigma,I)}, x) = ([\![E]\!]^+_{(\Sigma,I)}, x) \qquad ([\![(E)_A]\!]_{(\Sigma,I)}, x) = (([\![E]\!]_{(\Sigma,I)})_A, x)$$

and

$$([\![E \cdot F]\!]_{(\Sigma,I)}, x) = \sum_{\substack{y,z\in\mathbb{M}^+(\Sigma,I) \\ x=yz}} ([\![E]\!]_{(\Sigma,I)}, y) \cdot ([\![F]\!]_{(\Sigma,I)}, z)$$

for $A \subseteq \Sigma$ and $x \in \mathbb{M}^+(\Sigma, I)$. Usually, the independence alphabet (Σ, I) will be clear from the context. Therefore, we write $[\![E]\!]_{\mathrm{T}}$ for $[\![E]\!]_{(\Sigma,I)}$ and call it the *trace semantics* of E, and $[\![E]\!]_{\mathrm{W}}$ for $[\![E]\!]_{(\Sigma,\emptyset)}$, the *word semantics* of E.

With these notions, we have the following theorem by Schützenberger.

Theorem 2.1 ([13, 7]). *Let S be a semiring, Σ an alphabet, and $s \in S\langle\langle \Sigma^+ \rangle\rangle$. Then s is the word behavior of some weighted trace automaton iff it is the word semantics of some expression.*

Remark 2.1. Schützenberger [13] considers only the case of the semiring of integers, the general result can be found in Eilenberg's book [7]. Both these authors deal with formal power series over the free monoid Σ^*, i.e., also include the empty word. But the result holds likewise for the free semigroup of nonempty finite words. This follows easily from the following observations (with the obvious definitions [3, 5]).

1. A mapping $s : \mathbb{M}(\Sigma, I) \to S$ is recognizable in the sense of [13, 3] iff $s \restriction \mathbb{M}^+(\Sigma, I) \in S\langle\langle \mathbb{M}^+(\Sigma, I) \rangle\rangle$ is the trace behavior of some weighted trace automaton.

2. A mapping $s : \mathbb{M}(\Sigma, I) \to S$ is rational in the sense of [13, 3] iff $s \restriction (\mathbb{M}^+(\Sigma, I)$ is the trace semantics of some expression. The only difficulty in the inductive verification of this claim concerns multiplication. This problem can be solved since
$$(s \cdot t) \restriction \mathbb{M}^+(\Sigma, I) = (s, [\varepsilon]) \cdot t' + s' \cdot (t, [\varepsilon]) + s' \cdot t'$$
holds for all $s, t : \mathbb{M}(\Sigma, I) \to S$ with $s' = s \restriction \mathbb{M}^+(\Sigma, I)$ and $t' = t \restriction \mathbb{M}^+(\Sigma, I)$.

In addition, Schützenberger and Eilenberg do not allow the operation $(\ )_A$ in their expressions. Thus, in one sense, their result is stronger: any weighted automaton can be translated into an equivalent expression that does not use the operation $(\ )_A$. On the other hand, it follows from [7, Prop. VI.7.1] that even with the operation $(\ )_A$, we can only describe behaviors of weighted automata since the language $\{u \in \Sigma^+ \mid \mathrm{alph}(u) = A\}$ is regular.

Note that the mapping $\varphi : \Sigma^+ \to \mathbb{M}^+(\Sigma, I) : u \mapsto [u]$ is a semigroup homomorphism. From φ, we define another mapping $\overline{\varphi} : S\langle\langle \Sigma^+ \rangle\rangle \to S\langle\langle \mathbb{M}^+(\Sigma, I) \rangle\rangle$ setting (for all $x \in \mathbb{M}^+(\Sigma, I)$)
$$(\overline{\varphi}(s), x) = \sum_{u \in \varphi^{-1}(x)} (s, u) \ .$$

Then direct calculations show that $\overline{\varphi}$ commutes with the operations $+$, $\cdot$, $(\ )_A$, and $^+$. By induction, it follows that the word and the trace semantics of an expression are closely related:

Proposition 2.1. *Let S be some semiring, (Σ, I) an independence alphabet, and E an expression. Then $\overline{\varphi}([\![E]\!]_{\mathrm{W}}) = [\![E]\!]_{\mathrm{T}}$, i.e., for every $u \in \Sigma^+$, we have $([\![E]\!]_{\mathrm{T}}, [u]) = \sum_{v \in [u]} ([\![E]\!]_{\mathrm{W}}, v)$.*

A language $K \subseteq \mathbb{M}^+(\Sigma, I)$ is *mono-alphabetic* if $\mathrm{alph}(x) = \mathrm{alph}(y)$ for every $x, y \in K$; a formal power series $t \in S\langle\langle \mathbb{M}^+(\Sigma)\rangle\rangle$ is *mono-alphabetic* if its support $\{x \in \mathbb{M}^+(\Sigma, I) \mid (t, x) \neq 0\}$ is mono-alphabetic.

A set $B \subseteq \Sigma$ is *I-connected*, if $(B, D \cap B^2)$ is a connected graph; a word $u \in \Sigma^+$ is I-connected if $\mathrm{alph}(u)$ is I-connected; a language $L \subseteq \Sigma^+$ is I-connected if any of its elements is I-connected. Finally, a formal power series $t \in S\langle\langle \mathbb{M}^+(\Sigma)\rangle\rangle$ is I-connected if its support is I-connected.

Droste & Gastin [3, page 52] consider mc-rational and c-rational formal power series that are the semantics of mc-rational and c-rational expressions defined as follows: A *c-rational expression (over (Σ, I))* is an expression E not using the unary operation $(\,)_A$ such that $[\![F]\!]_{\mathrm{T}}$ is I-connected for all sub-expressions F^+ of E. If, in addition, $[\![F]\!]_{\mathrm{T}}$ is mono-alphabetic and I-connected for all subexpressions F^+, the expression is *mc-rational*.

3 From expressions to automata

Given an expression E over (Σ, I), we want to construct a weighted trace automaton $\mathcal{A}$ with $[\![E]\!]_{\mathrm{T}} = \|\mathcal{A}\|_{\mathrm{T}}$. Recall that $[\![E]\!]_{\mathrm{T}} = \overline{\varphi}([\![E]\!]_{\mathrm{W}})$ by Prop. 2.1. Hence, we will first describe a condition on a series $s \in S\langle\langle \Sigma^+\rangle\rangle$ implying that $\overline{\varphi}(s)$ is the trace behavior of some weighted trace automaton over (Σ, I) (Prop. 3.1). Droste & Gastin [3, Example 39] showed that the fps $[\![1a + 1b]\!]_{\mathrm{T}}$ over the semiring of natural numbers is not the behavior of any weighted trace automaton provided $(a, b) \in I$. Hence, we cannot hope for $[\![E]\!]_{\mathrm{T}}$ to satisfy the condition for each and every expression E, but we prove it for mild extensions of c-rational and mc-rational expressions.

3.1 The condition

Let (Σ, I_Σ) and (Γ, I_Γ) be independence alphabets. A function $\pi : \Gamma \to \Sigma$ is a *projection of independence alphabets* if $(A, B) \in I_\Gamma \iff (\pi(A), \pi(B)) \in I_\Sigma$ for all $A, B \in \Gamma$.

Lemma 3.1. *Let S be a commutative semiring and $\pi : (\Gamma, I_\Gamma) \to (\Sigma, I_\Sigma)$ be a projection of independence alphabets. Furthermore, let $K \subseteq \Gamma^+$ be an I_Γ-closed regular language and $c : \Gamma^+ \to (S, \cdot)$ be a homomorphism. Then there exists a weighted trace automaton $\mathcal{A}$ such that we have for all $u \in \Sigma^+$*

$$(\|\mathcal{A}\|_{\mathrm{W}}, u) = \sum (c(U) \mid U \in K \cap \pi^{-1}(u)) \ .$$

Proof. Let $\mathcal{B} = (Q, \Gamma, \iota, \delta, F)$ denote the minimal deterministic finite automaton with $\mathcal{L}(\mathcal{B}) = K$. Then we have $\delta(q, AB) = \delta(q, BA)$ for every $q \in Q$ and $A, B \in \Gamma$ with $(A, B) \in I_\Gamma$ since $\mathcal{B}$ is minimal and its language K is I_Γ-closed.

From $\mathcal{B}$, we construct a weighted finite automaton $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ on the set of states Q of $\mathcal{B}$ setting

- $\lambda(q) = 1$ for $q = \iota$ and $\lambda(q) = 0$ for $q \neq \iota$

- $\gamma(q) = 1$ for $q \in F$ and $\gamma(q) = 0$ for $q \notin F$
- $\mu(a)_{q_1,q_2} = \sum(c(A) \mid A \in \Gamma \text{ with } \pi(A) = a \text{ and } q_2 = \delta(q_1, A))$ for every $a \in \Sigma$ and $q_1, q_2 \in Q$.

We first verify that $\mathcal{A}$ is indeed a weighted trace automaton, i.e., that $\mu(ab) = \mu(ba)$ for all $a, b \in \Sigma$ with $(a, b) \in I_\Sigma$. For this, let $q_1, q_3 \in Q$. Then

$$\begin{aligned}
\mu(ab)_{q_1,q_3} &= \sum_{q_2 \in Q} \mu(a)_{q_1,q_2} \cdot \mu(b)_{q_2,q_3} \\
&= \sum_{q_2 \in Q} \sum \left(c(A) \cdot c(B) \;\middle|\; \begin{array}{l} A, B \in \Gamma, \pi(A) = a, \pi(B) = b, \\ q_2 = \delta(q_1, A), q_3 = \delta(q_2, B) \end{array} \right) \\
&\overset{(i)}{=} \sum \left(c(A) \cdot c(B) \;\middle|\; \begin{array}{l} A, B \in \Gamma, \pi(A) = a, \pi(B) = b, \\ q_3 = \delta(q_1, AB) \end{array} \right) \\
&\overset{(ii)}{=} \sum \left(c(B) \cdot c(A) \;\middle|\; \begin{array}{l} A, B \in \Gamma, \pi(A) = a, \pi(B) = b, \\ q_3 = \delta(q_1, BA) \end{array} \right) \\
&= \mu(ba)_{q_1,q_3} \,.
\end{aligned}$$

Equation (i) holds since $\mathcal{B}$ is a deterministic automaton. Regarding equation (ii), note that $(A, B) \in I_\Gamma$ since $(a, b) \in I_\Sigma$ and since π is a projection of independence alphabets. Then equation (ii) follows since the semiring is commutative and $\delta(q, AB) = \delta(q, BA)$ for every $q \in Q$ and $A, B \in \Gamma$ with $(A, B) \in I_\Gamma$.

It remains to verify $(\|\mathcal{A}\|_W, u) = \sum(c(U) \mid U \in K \cap \pi^{-1}(u))$. So let $u = a_1 \dots a_n \in \Sigma^+$ with $a_i \in \Sigma$. Then we have (where all products stretch over the set of indices $1 \le i \le n$)

$$\begin{aligned}
(\|\mathcal{A}\|_W, u) &= \sum_{p,q \in Q} \lambda(p) \cdot \mu(u)_{p,q} \cdot \gamma(q) = \sum_{f \in F} \mu(u)_{\iota,f} \\
&= \sum \Big(\prod \mu(a_i)_{q_{i-1},q_i} \mid q_0 = \iota, q_1, q_2, \dots, q_{n-1} \in Q, q_n \in F \Big) \\
&= \sum_{\substack{q_0 = \iota \\ q_1, \dots, q_{n-1} \in Q \\ q_n \in F}} \prod \sum (c(A_i) \mid A_i \in \Gamma, \pi(A_i) = a_i, q_{i+1} = \delta(q_i, A_i)) \\
&= \sum \left(\prod c(A_i) \;\middle|\; \begin{array}{l} \iota = q_0, q_1, \dots, q_{n-1} \in Q, q_n \in F, \\ \text{for } 1 \le i \le n : A_i \in \Gamma, \pi(A_i) = a_i, q_{i+1} = \delta(q_i, A_i) \end{array} \right) \\
&= \sum (c(U) \mid U \in \Gamma^+ \text{ with } \delta(\iota, U) \in F \text{ and } \pi(U) = u) \\
&= \sum (c(U) \mid U \in K \cap \pi^{-1}(u)) \,.
\end{aligned}$$

□

For a language $L \subseteq \Gamma^+$, we write $[L]$ for the set $\bigcup_{U \in L}[U] = \{V \in \Gamma^+ \mid \exists U \in L : U \sim V\}$.

Proposition 3.1. *Let S be a commutative semiring and $\pi : (\Gamma, I_\Gamma) \to (\Sigma, I_\Sigma)$ be a projection of independence alphabets, let $c : \Gamma^+ \to (S, \cdot)$ be a homomorphism, $L \subseteq \Gamma^+$ a language, and $s \in S\langle\langle \Sigma^+ \rangle\rangle$ a fps such that*

(T1) $[L]$ *is regular,*

(T2) $(s, u) = \sum(c(U) \mid U \in L \cap \pi^{-1}(u))$ *for all* $u \in \Sigma^+$, *and*

(T3) $\sum(c(U) \mid U \in [L] \cap \pi^{-1}(u)) = \sum(c(V) \mid V \in L \cap [\pi^{-1}(u)])$ *for all* $u \in \Sigma^+$.

Then there exists a weighted trace automaton $\mathcal{A}$ over (Σ, I_Σ) such that $\|\mathcal{A}\|_{\mathrm{T}} = \overline{\varphi}(s)$.

Proof. Note that the language $[L]$ is I_Γ-closed. Hence we can apply Lemma 3.1 which yields a weighted trace automaton $\mathcal{A}$. Then we have for any $u \in \Sigma^+$

$$
\begin{aligned}
(\|\mathcal{A}\|_{\mathrm{T}}, [u]) &= (\|\mathcal{A}\|_{\mathrm{W}}, u) \\
&= \sum(c(U) \mid U \in [L] \cap \pi^{-1}(u)) && \text{by Lemma 3.1} \\
&= \sum(c(V) \mid V \in L \cap [\pi^{-1}(u)]) && \text{by } (T3) \\
&= \sum_{v \sim u} \sum(c(V) \mid V \in L \cap \pi^{-1}(v)) && \text{since } [\pi^{-1}(u)] = \pi^{-1}([u]) \\
&= \sum_{v \sim u} (s, v) && \text{by } (T2) \\
&= (\overline{\varphi}(s), u) && \text{by definition of } \overline{\varphi}.
\end{aligned}
$$

□

3.2 c-expressions

Throughout this section, we fix an independence alphabet (Σ, I_Σ). Let CONN denote the set of I_Σ-connected subsets of Σ.

For $s \in S\langle\langle \Sigma^+ \rangle\rangle$ and $t \in S\langle\langle \mathbb{M}^+(\Sigma, I_\Sigma) \rangle\rangle$, define

$$
s^{\mathrm{c}+} = \left[\sum_{A \in \mathrm{CONN}} (s)_A \right]^+ \quad \text{and} \quad t^{\mathrm{c}+} = \left[\sum_{A \in \mathrm{CONN}} (t)_A \right]^+ .
$$

Suppose that, for all $x \in \mathbb{M}^+(\Sigma, I_\Sigma)$ with $(s, x) \neq 0$, we have $\mathrm{alph}(x) \in \mathrm{CONN}$. Then it is immediate that $s^+ = s^{\mathrm{c}+}$.

Definition 3.1. *A* c-expression *is a term using the constants ka for $k \in S$ and $a \in \Sigma$, the binary operations $+$ and $\cdot$, and the unary operation ${}^{\mathrm{c}+}$.*

Since ${}^{\mathrm{c}+}$ can be expressed in terms of the operations of expressions, c-expressions are special expressions and we will handle them as expressions. In particular, the word and trace semantics of c-expressions are inherited from those of expressions.

Remark 3.1. Let E be some c-rational expression (i.e., $[\![E]\!]_{\mathrm{T}}$ is a c-rational formal power series as defined by Droste & Gastin [3]). Replacing, in E, any occurrence of $^+$ with $^{\mathrm{c}+}$ then results in an equivalent c-expression E', i.e., $[\![E]\!]_{\mathrm{T}} = [\![E']\!]_{\mathrm{T}}$. Hence, any c-rational formal power series is the trace semantics of some c-expression.

The rest of this section is devoted to the proof of the following

Theorem 3.1 (cf. [3, Thm. 1(c)]). *Let S be a commutative and idempotent semiring and E a c-expression. Then there exists a weighted trace automaton $\mathcal{A}$ over (Σ, I_Σ) such that $[\![E]\!]_{\mathrm{T}} = \|\mathcal{A}\|_{\mathrm{T}}$.*

The proof will be based on Prop. 3.1. More precisely, we will first replace, in the expression E, every appearance of ka with a new letter (k, a) and $^{\mathrm{c}+}$ with $^+$. This results in a rational expression whose language L, together with the functions π and c given by $\pi(k, a) = a$ and $c(k, a) = k$, satisfies (T1-3) for $s = [\![E]\!]_{\mathrm{W}}$ (see below). Then, from Prop. 3.1, we obtain a weighted trace automaton $\mathcal{A}$ with $\|\mathcal{A}\|_{\mathrm{T}} = \overline{\varphi}([\![E]\!]_{\mathrm{W}})$ which equals $[\![E]\!]_{\mathrm{T}}$ by Prop. 2.1.

3.2.1 The construction

Since we want to use Prop. 3.1, we have to construct an independence alphabet (Γ, I_Γ), a projection of independence alphabets $\pi : (\Gamma, I_\Gamma) \to (\Sigma, I_\Sigma)$, a homomorphism $c : \Gamma^+ \to (S, \cdot)$, and a language $L \subseteq \Gamma^+$ such that (T1-3) hold.

- Γ is the set of all pairs $(k, a) \in S \times \Sigma$ such that the constant ka appears in the c-expression E.
- For $(k, a), (\ell, b) \in \Gamma$, we set $((k, a), (\ell, b)) \in I_\Gamma$ iff $(a, b) \in I_\Sigma$.
- For $(k, a) \in \Gamma$, let $\pi(k, a) = a$. This defines a projection of independence alphabets $\pi : (\Gamma, I_\Gamma) \to (\Sigma, I_\Sigma)$.
- Let $c : \Gamma^+ \to (S, \cdot)$ be the homomorphism defined by $c(k, a) = k$ for $(k, a) \in \Gamma$.

From a c-expression, we define inductively a language over Γ as follows:

$$[\![ka]\!]' = \{(k, a)\} \qquad [\![E_1 + E_2]\!]' = [\![E_1]\!]' \cup [\![E_2]\!]'$$
$$[\![E_1 \cdot E_2]\!]' = [\![E_1]\!]'[\![E_2]\!]' \qquad [\![E_1^{\mathrm{c}+}]\!]' = \{U \in [\![E_1]\!]' \mid U \text{ is connected}\}^+$$

The language $L \subseteq \Gamma^+$ that we need for the application of Prop. 3.1 is $L = [\![E]\!]'$.

3.2.2 Verification of (T1-3)

Note that the language L is constructed from the singletons by union, concatenation, intersection with the set of connected words, and iteration $^+$. In this construction, iteration is only applied to connected languages. Hence, by [12], the language $[L]$ is regular. This verifies (T1).

Property (T2) is verified inductively along the construction of the c-expression E, i.e., we prove

$$([\![E_1]\!]_{\mathrm{W}}, u) = \sum (c(U) \mid U \in [\![E_1]\!]' \cap \pi^{-1}(u)) \tag{2}$$

for all $u \in \Sigma^+$ and for all sub-expressions E_1 of E:

- for $E_1 = ka$, we have $([\![E_1]\!]_{\mathrm{W}}, u) = k$ for $u = a$ and $([\![E_1]\!]_{\mathrm{W}}, u) = 0$ otherwise. On the other hand, $[\![E_1]\!]' = \{(k, a)\}$ proving Eq. (2).
- Provided Eq. (2) holds for the c-expressions E_1 and E_2, we obtain

$$\begin{aligned}([\![E_1 + E_2]\!]_{\mathrm{W}}, u) &= ([\![E_1]\!]_{\mathrm{W}}, u) + ([\![E_2]\!]_{\mathrm{W}}, u) \\ &= \sum (c(U) \mid U \in [\![E_1]\!]', \pi(U) = u) \\ &\quad + \sum (c(U) \mid U \in [\![E_2]\!]', \pi(U) = u)\end{aligned}$$

since the semiring S is idempotent, this last expression equals

$$\begin{aligned}&= \sum (c(U) \mid U \in [\![E_1]\!]' \cup [\![E_2]\!]', \pi(U) = u) \\ &= \sum (c(U) \mid U \in [\![E_1 + E_2]\!]', \pi(U) = u).\end{aligned}$$

Furthermore,

$$\begin{aligned}([\![E_1 \cdot E_2]\!]_{\mathrm{W}}, u) &= \sum (([\![E_1]\!]_{\mathrm{W}}, v) \cdot ([\![E_2]\!]_{\mathrm{W}}, w) \mid v, w \in \Sigma^+, u = vw) \\ &= \sum (c(V) \cdot c(W) \mid V \in [\![E_1]\!]', W \in [\![E_2]\!]', u = \pi(V)\pi(W)) \\ &\overset{(*)}{=} \sum (c(U) \mid U \in [\![E_1 \cdot E_2]\!]', u = \pi(U)) \ .\end{aligned}$$

Regarding the equation (*), note that $(V, W) \mapsto VW$ is a surjection from the set of pairs $\{(V, W) \in [\![E_1]\!]' \times [\![E_2]\!]' \mid \pi(V)\pi(W) = u\}$ onto the set of words $\{U \in [\![E_1 \cdot E_2]\!]' \mid \pi(U) = u\} : (V, W) \mapsto VW$ and that $c(VW) = c(V)c(W)$. Then (*) holds since the semiring S is idempotent.

- Now suppose Eq. (2) holds for the c-expression F and let $u \in \Sigma^+$. Then we have

$$\begin{aligned}([\![F^{\mathrm{c}+}]\!]_{\mathrm{W}}, u) &= \left(\left[\sum_{A \in \mathrm{CONN}} ([\![F]\!]_{\mathrm{W}})_A \right]^+ , u \right) \\ &= \sum \left(\prod_{1 \le j \le i} ([\![F]\!]_{\mathrm{W}}, u_j) \;\middle|\; \begin{array}{l} 1 \le i \le |u|, u = u_1 u_2 \ldots u_i, \\ u_1, \ldots, u_i \in \Sigma^+ \text{ with} \\ \mathrm{alph}(u_j) \in \mathrm{CONN} \end{array} \right) \\ &= \sum_{\substack{1 \le i \le |u|, u = u_1 u_2 \ldots u_i, \\ u_1, \ldots, u_i \in \Sigma^+, \mathrm{alph}(u_j) \in \mathrm{CONN}}} \prod_{1 \le j \le i} \sum (c(U_j) \mid U_j \in [\![F]\!]', \pi(U_j) = u_j)\end{aligned}$$

$$= \sum_{\substack{1\le i\le|u|, u=u_1u_2\dots u_i,\\ u_1,\dots,u_i\in\Sigma^+, \mathrm{alph}(u_j)\in\mathrm{CONN}}} \sum (c(U_1U_2\dots U_i) \mid U_j \in [\![F]\!]', \pi(U_j) = u_j)$$

$$= \sum \left(c(U_1U_2\dots U_i) \;\middle|\; \begin{array}{l} 1 \le i \le |u|, u = u_1u_2\dots u_i, \\ u_1,\dots,u_i \in \Sigma^+, \mathrm{alph}(u_j) \in \mathrm{CONN} \\ U_j \in [\![F]\!]', \pi(U_j) = u_j \end{array} \right)$$

since $U_j \in \Gamma^+$ is I_Γ-connected iff $\pi(U_j) \in \Sigma^+$ is I_Σ-connected, we can continue

$$= \sum \left(c(U_1U_2\dots U_i) \;\middle|\; \begin{array}{l} 1 \le i \le |u| \\ U_1,\dots,U_i \in [\![F]\!]' \ I_\Gamma\text{-connected} \\ \pi(U_1U_2\dots U_i) = u \end{array} \right)$$

$$\overset{(*)}{=} \sum \left(c(U) \mid U \in [\![F^{c+}]\!]', \pi(U) = u \right).$$

Here, the equation $(*)$ holds since $(U_1, U_2, \dots, U_i) \mapsto (U_1U_2\dots U_i)$ is a surjection from the set

$$\left\{ (U_1,\dots,U_i) \;\middle|\; \begin{array}{l} 1 \le i \le |u| \\ U_1,\dots,U_i \in [\![F]\!]' \ I_\Gamma\text{-connected} \\ \pi(U_1U_2\dots U_i) = u \end{array} \right\}$$

onto the set

$$\{U \in [\![F^{c+}]\!]' \mid \pi(U) = u\}.$$

This finishes the verification of (T2).

Finally, we verify (T3). So let $u \in \Sigma^+$ and $V \in L \cap [\pi^{-1}(u)]$. Then there exists $U \in \pi^{-1}(u)$ with $V \sim U$ implying $U \in [L] \cap \pi^{-1}(u)$. Hence there is a function $f_u : L \cap [\pi^{-1}(u)] \to [L] \cap \pi^{-1}(u)$ with $f_u(V) \sim V$ and therefore $c(f_u(V)) = c(V)$ since $(S, \cdot)$ is commutative. This function is even surjective: if $U \in [L] \cap \pi^{-1}(u)$, then there exists at least one word $V \in L$ with $U \sim V$ and therefore $V \in L \cap [\pi^{-1}(u)]$. Since π is a projection of independence alphabets, this implies $\pi(V) \sim \pi(U) = u$. Hence we have $f_u(V) \sim V \sim U$ and $\pi(f_u(V)) = u = \pi(U)$ which implies $U = f_u(V)$. Since the semiring S is idempotent, this ensures (T3).

Since we successfully verified (T1-3), Thm. 3.1 follows from Prop. 3.1.

3.3 mc-expressions

Again, we fix an independence alphabet (Σ, I_Σ) and let CONN denote the set of I_Σ-connected subsets of Σ.

For $s \in S\langle\langle \Sigma^+ \rangle\rangle$ and $t \in S\langle\langle \mathbb{M}^+(\Sigma, I_\Sigma) \rangle\rangle$, define

$$s^{\mathrm{mc}+} = \sum_{A\in\mathrm{CONN}} [(s)_A]^+ \text{ and } t^{\mathrm{mc}+} = \sum_{A\in\mathrm{CONN}} [(t)_A]^+ .$$

Suppose there exists $A \in \mathrm{CONN}$ such that, for all $x \in \mathbb{M}^+(\Sigma, I_\Sigma)$ with $(s, x) \neq 0$, we have $\mathrm{alph}(x) = A$. Then it is immediate that $s^+ = s^{\mathrm{mc}+}$.

Definition 3.2. *An* mc-expression *is a term using the constants ka for $k \in S$ and $a \in \Sigma$, the binary operations $+$ and $\cdot$, and the unary operation* $^{\mathrm{mc}+}$.

Since $^{\mathrm{mc}+}$ can be expressed in terms of the operations of expressions, mc-expressions are special expressions and we will handle them as expressions. In particular, the word and trace semantics of mc-expressions are inherited from those of expressions.

Remark 3.2. Let E be some mc-rational expression (i.e., $[\![E]\!]_{\mathrm{T}}$ is an mc-rational formal power series as defined by Droste & Gastin [3]). Replacing, in E, any occurrence of $^+$ with $^{\mathrm{mc}+}$ results in an equivalent mc-expression E', i.e., $[\![E]\!]_{\mathrm{T}} = [\![E']\!]_{\mathrm{T}}$. Hence, any mc-rational formal power series is the trace semantics of some mc-expression.

The rest of this section is devoted to the proof of the following

Theorem 3.2 (cf. [3, Thm. 1(b)]). *Let S be some commutative semiring and E some mc-expression. Then there exists a weighted trace automaton $\mathcal{A}$ such that $[\![E]\!]_{\mathrm{T}} = \|\mathcal{A}\|_{\mathrm{T}}$.*

3.3.1 The construction

The first idea is to proceed analogously to the proof of Thm. 3.1, i.e., to first replace, in the mc-expression E, every appearance of ka with a new letter (k,a) and $^{\mathrm{mc}+}$ with $^+$. The resulting expression describes a language L. Furthermore, we would set $\pi(k,a) = a$ and $c(k,a) = k$ for $(k,a) \in \Gamma$. Since the semiring S is not assumed to be idempotent anymore, verification of (T2) with $s = [\![E]\!]_{\mathrm{W}}$ causes problems that are best explained by the following two examples using the semiring $\mathcal{N} = (\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers.

- The mc-expression $E = 1a + 1a$ would be transformed into the rational expression $(1,a) + (1,a)$, i.e., $L = \{(1,a)\}$. With $u = a$, the left hand side in (T2) then equals 2, the right-hand side is just 1.
- The mc-expression $E = ((1a)^{\mathrm{mc}+})^{\mathrm{mc}+}$ would be transformed into $((1,a)^+)^+$, i.e., $L = \{(1,a)\}^+$. Note that $[\![(1a)^{\mathrm{mc}+}]\!]_{\mathrm{W}}$ is the characteristic function of $\{a\}^+$, hence

  $$([\![E]\!]_{\mathrm{W}}, aa) = ([\![(1a)^{\mathrm{mc}+}]\!]_{\mathrm{W}}, aa) + ([\![(1a)^{\mathrm{mc}+}]\!]_{\mathrm{W}}, a) \cdot ([\![(1a)^{\mathrm{mc}+}]\!]_{\mathrm{W}}, a) = 2 \,.$$

  On the other hand, the right-hand side of (T2) yields 1 (with $u = aa$).

The sole reason for these problems is that the Boolean semiring is idempotent while the semiring S can be arbitrary. The first of these problems can be solved by replacing the constants in E with pairwise distinct new letters. A solution to the second problem is based on the observation that

$$[\![E^+]\!]_{\mathrm{W}} = [\![E + (E \cdot E)^+ + (E \cdot E)^+ \cdot E]\!]_{\mathrm{W}} \,.$$

To perform this programme formally, we define a relation Red between mc-expressions E over Σ, alphabets Γ, languages $L \subseteq \Gamma^+$, functions $\pi : \Gamma \to \Sigma$, and homomorphisms $c : \Gamma^+ \to (S, \cdot)$. Set $(E, \Gamma, L, \pi, c) \in \mathrm{Red}$ iff

1. $E = ka$, $\Gamma = \{\bot\}$ for some letter $\bot$, $L = \{\bot\}$, $\pi(\bot) = a$, and $c(\bot) = k$, or
2. there exist $(E_i, \Gamma_i, L_i, \pi_i, c_i) \in \mathrm{Red}$ for $i = 0, 1$ with $\Gamma_0 \cap \Gamma_1 = \emptyset$, $\Gamma = \Gamma_0 \cup \Gamma_1$, $\pi = \pi_0 \cup \pi_1$, $c{\restriction}\Gamma = c_0{\restriction}\Gamma_0 \cup c_1{\restriction}\Gamma_1$, and one of the following holds
   a) $E = E_0 + E_1$ and $L = L_0 \cup L_1$,
   b) $E = E_0 \cdot E_1$ and $L = L_0 \cdot L_1$, or
   c) $E = E_0^{\mathrm{mc}+}$, $E_0 = E_1$, and

$$L = \bigcup_{A \in \mathrm{CONN}} L_1^A \cup (L_1^A \cdot L_0^A)^+ \cup (L_1^A \cdot L_0^A)^+ \cdot L_1^A$$

where L_i^A is the set of words $U \in L_i$ with $\pi(\mathrm{alph}(U)) = A$.

Let $(E, \Gamma, L, \pi, c) \in \mathrm{Red}$. Then one can show by induction that $L \subseteq \Gamma^+$ is a regular language, the only nontrivial case is $E = E_0^+$ where one has to observe that L^A is regular as soon as L is regular. Furthermore, the binary relation

$$I_\Gamma = \{(A, B) \in \Gamma \mid (\pi(A), \pi(B)) \in I_\Sigma\}$$

is the only independence relation on Γ such that $\pi : (\Gamma, I_\Gamma) \to (\Sigma, I_\Sigma)$ is a projection of independence alphabets. In the following, we will always assume Γ to be equipped with this independence relation.

3.3.2 Verification of (T1-3)

Lemma 3.2. *For $(E, \Gamma, L, \pi, c) \in \mathrm{Red}$, the language $[L] \subseteq \Gamma^+$ is regular.*

Proof. We proceed by induction along the construction of the mc-expression E. By [12], the base case $E = ka$ as well as the inductive arguments for the cases $E = E_0 + E_1$ and $E = E_0 \cdot E_1$ are immediate. So assume $E = E_0^{\mathrm{mc}+}$, $E_0 = E_1$, and $(E_i, \Gamma^i, L_i, \pi_i, c_i) \in \mathrm{Red}$. Let $U \in L_1^A \cdot L_0^A$ for some $A \in \mathrm{CONN}$. Then $U = V_1 V_0$ for some words $V_i \in L_i^A$. Hence $\pi(\mathrm{alph}(V_i)) = A$ implying $\pi(\mathrm{alph}(U)) = A$. Since $A \in \mathrm{CONN}$, the set $\mathrm{alph}(U)$ is I_Γ-connected. Hence the language $L_1^A \cdot L_0^A$ is connected. Now, from [12], we obtain that $[L]$ is regular. □

Lemma 3.3. *For $(E, \Gamma, L, \pi, c) \in \mathrm{Red}$, the following holds for all $u \in \Sigma^+$:*

$$([\![E]\!]_{\mathrm{W}}, u) = \sum (c(U) \mid U \in L \cap \pi^{-1}(u)) \ .$$

Proof. The lemma is shown by induction on the construction of E. The base case $E = ka$ is obvious. Now suppose that the lemma has been shown for the tuples $(E_i, \Gamma_i, L_i, \pi_i, c_i) \in \mathrm{Red}$ $(i = 0, 1)$. Furthermore, assume $\Gamma_0 \cap \Gamma_1 = \emptyset$, $\Gamma = \Gamma_0 \cup \Gamma_1$, $\pi = \pi_0 \cup \pi_1$, and $c{\restriction}\Gamma = c_0{\restriction}\Gamma_0 \cup c_1{\restriction}\Gamma_1$.

First suppose $E = E_0 + E_1$ and $L = L_0 \cup L_1$. Then we have

$$\begin{aligned}([\![E]\!]_{\mathrm{W}}, u) &= ([\![E_0]\!]_{\mathrm{W}}, u) + ([\![E_1]\!]_{\mathrm{W}}, u) \\ &= \sum(c_0(U) \mid U \in L_0 \cap \pi_0^{-1}(u)) + \sum(c_1(U) \mid U \in L_1 \cap \pi_1^{-1}(u))\end{aligned}$$

Since the alphabets Γ_0 and Γ_1 are disjoint, so are the languages L_0 and L_1. Furthermore, c_i agrees with c on Γ_i^+ and similarly for π_i. Hence we can continue

$$\begin{aligned}&= \sum(c(U) \mid U \in (L_0 \cup L_1) \cap \pi^{-1}(u)) \\ &= \sum(c(U) \mid U \in L \cap \pi^{-1}(u)) \,.\end{aligned}$$

Next let $E = E_0 \cdot E_1$ and $L = L_0 \cdot L_1$. Then we have

$$\begin{aligned}([\![E]\!]_{\mathrm{W}}, u) &= \sum_{u=vw} ([\![E_0]\!]_{\mathrm{W}}, v) \cdot ([\![E_1]\!]_{\mathrm{W}}, w) \\ &= \sum_{u=vw} \left(\begin{array}{c} \sum(c_0(V) \mid V \in L_0 \cap \pi_0^{-1}(v)) \\ \cdot \sum(c_1(W) \mid W \in L_1 \cap \pi_1^{-1}(w)) \end{array} \right) \\ &= \sum \left(c_0(V) \cdot c_1(W) \,\middle|\, \begin{array}{l} u = vw, V \in L_0 \cap \pi_0^{-1}(v), \\ W \in L_1 \cap \pi_1^{-1}(w) \end{array} \right)\end{aligned}$$

Since the alphabets Γ_0 and Γ_1 are disjoint, every word U from $L = L_0 \cdot L_1$ has a unique factorization VW into factors from L_0 and L_1, resp. Hence we can continue

$$= \sum(c(U) \mid U \in L \cap \pi^{-1}(u)) \,.$$

Finally assume $E = E_0^{\mathrm{mc}+}$, $E_0 = E_1$, and $L = \bigcup_{A \in \mathrm{CONN}} L_1^A \cup (L_1^A \cdot L_0^A)^+ \cup (L_1^A \cdot L_0^A)^+ \cdot L_1^A$. Now let $u \in \Sigma^+$ with $B = \mathrm{alph}(u)$. If $B \notin \mathrm{CONN}$, then both sides of the equation from the lemma yield 0. So assume $B \in \mathrm{CONN}$. Then $([\![E]\!]_{\mathrm{W}}, u) = ((([\![E_0]\!]_{\mathrm{W}})_B)^+, u)$. In the following equations, we write π_j for $\pi_{j \bmod 2}$ and similarly L_j for $L_{j \bmod 2}$ for any $j \geq 1$. Then we get

$$\begin{aligned}([\![E]\!]_{\mathrm{W}}, u) &= \sum_{1 \leq i \leq |u|} \sum_{\substack{u = u_1 \ldots u_i \\ \pi_j(\mathrm{alph}(u_j)) = B}} \prod_{1 \leq j \leq i} ([\![E_0]\!]_{\mathrm{W}}, u_j) \\ &= \sum_{1 \leq i \leq |u|} \sum_{\substack{u = u_1 \ldots u_i \\ \pi_j(\mathrm{alph}(u_j)) = B}} \prod_{1 \leq j \leq i} \sum \left(c_j(U_j) \mid U_j \in L_j \cap \pi_j^{-1}(u_j) \right) \\ &= \sum \left(c(U_1) \cdot c(U_2) \cdot \ldots c(U_i) \,\middle|\, \begin{array}{l} 1 \leq i \leq |u|, u = u_1 \ldots u_i, \\ \text{for all } 1 \leq j \leq i : \\ \quad \pi(\mathrm{alph}(u_j)) = B \\ \quad U_j \in L_j \cap \pi^{-1}(u_j) \end{array} \right)\end{aligned}$$

where we used that c and π coincide with c_i and π_i on Γ_i^+.

Since the alphabets Γ_0 and Γ_1 are disjoint, every word U from $L \cap \pi^{-1}(u)$ has a unique factorization $U_1 U_2 \dots U_i$ into alternating factors from L_1^B and L_0^B and no factorization into alternating factors from L_1^A and L_0^A for $B \neq A \subseteq \Sigma$. Hence the above expression equals $\sum(c(U) \mid U \in L \cap \pi^{-1}(u))$. □

Lemma 3.4. *For* $(E, \Gamma, L, \pi, c) \in \mathrm{Red}$ *and* $u \in \Sigma^+$, *we have*

$$V_1, V_2 \in L \text{ and } V_1 \sim V_2 \implies V_1 = V_2 \ .$$

Proof. The lemma is shown by induction on the construction of E. The base case $E = ka$ is obvious. Now suppose that the lemma has been shown for the tuples $(E_i, \Gamma_i, L_i, \pi_i, c_i) \in \mathrm{Red}$ $(i = 0, 1)$. Furthermore, assume $\Gamma_0 \cap \Gamma_1 = \emptyset$, $\Gamma = \Gamma_0 \cup \Gamma_1$, $\pi = \pi_0 \cup \pi_1$, and $c|\Gamma = c_0|\Gamma_0 \cup c_1|\Gamma_1$

Now suppose $E = E_0 + E_1$ and $L = L_0 \cup L_1$. From $V_1 \sim V_2$, we obtain $\mathrm{alph}(V_1) = \mathrm{alph}(V_2)$. Since the languages L_1 and L_2 have disjoint alphabets, $V_1, V_2 \in L$ implies $V_1, V_2 \in L_i$ for $i = 0$ or for $i = 1$. Hence, by the induction hypothesis, $V_1 \sim V_2$ implies $V_1 = V_2$.

Next suppose $E = E_0 \cdot E_1$ and $L = L_0 \cdot L_1$. Then $V_1, V_2 \in L$ implies the existence of $V_i^j \in L_j$ for $j = 0, 1$ with $V_i = V_i^0 V_i^1$ for $i = 1, 2$. By disjointness of the alphabets, $V_1 \sim V_2$ implies $V_1^j \sim V_2^j$ for $j = 1, 2$. Hence, by the induction hypothesis, $V_i^1 = V_i^2$ and therefore $V_1 = V_2$.

Finally let $E = E_0^{\mathrm{mc}+}$, $E_0 = E_1$, and

$$L = \bigcup_{A \in \mathrm{CONN}} L_1^A \cup (L_1^A \cdot L_0^A)^+ \cup (L_1^A \cdot L_0^A)^+ \cdot L_1^A \ .$$

From $V_1 \in L$, we obtain $B = \pi(\mathrm{alph}(V_1)) \in \mathrm{CONN}$ and $V_1 = V_1^1 V_1^2 \dots V_1^{i_1}$ with $V_1^j \in L_{j \bmod 2}^B$ for all $1 \leq j \leq i_1$. From $V_1 \sim V_2$, we deduce $\mathrm{alph}(V_1) = \mathrm{alph}(V_2)$ and therefore $\pi(\mathrm{alph}(V_1)) = \pi(\mathrm{alph}(V_2))$. Hence $V_2 = V_2^1 V_2^2 \dots V_2^{i_2}$ with $V_1^j \in L_{j \bmod 2}^B$ for all $1 \leq j \leq i_2$.

For $B \subseteq \Sigma$ and $W \in \Gamma^+$, let $\mathrm{proj}_B(W)$ denote the projection of W to the letters from $\pi^{-1}(B)$, i.e., $\mathrm{proj}_B(W)$ is obtained from W by deleting all letters $\gamma \in \Gamma$ with $\pi(\gamma) \notin B$. Now assume that any two letters from $\emptyset \neq B \subseteq \Sigma$ are dependent. Then the same holds for $\pi^{-1}(B)$. Hence $V_1 \sim V_2$ implies $\mathrm{proj}_B(V_1) = \mathrm{proj}_B(V_2)$. Since $V_i^j \in L_{j \bmod 2}^A$, we have $\mathrm{proj}_B(V_i^j) \neq \varepsilon$ for all $\emptyset \neq B \subseteq A$. Since the independence alphabets are disjoint, this implies $i_1 = i_2$ and $\mathrm{proj}_B(V_1^j) = \mathrm{proj}_B(V_2^j)$ for all $1 \leq j \leq i_1$ and $\emptyset \neq B \subseteq A$ with $B \times B \subseteq D$. But this implies $V_1^j \sim V_2^j$ for all $1 \leq j \leq i_1$ and therefore, by the induction hypothesis, $V_1^j = V_2^j$. Hence, indeed, $V_1 = V_2$. □

Lemma 3.5. *For* $(E, \Gamma, L, \pi, c) \in \mathrm{Red}$ *and* $u \in \Sigma^+$, *we have*

$$\sum(c(U) \mid U \in [L] \cap \pi^{-1}(u)) = \sum(c(V) \mid V \in L \cap [\pi^{-1}(u)]) \ .$$

Proof. As in the verification of (T3) from Section 3.2 (page 402), there is a surjection $f_u : L \cap \pi^{-1}([u]) \to [L] \cap \pi^{-1}(u)$ with $U \sim f_u(U)$ and therefore $c(U) = c(f_u(U))$. Hence, by Lemma 3.4, f_u is injective and therefore a weight-preserving bijection. This implies the statement. □

Proof of Thm. 3.2. Inductively, one finds a tuple $(E, \Gamma, L, \pi, c) \in \mathrm{Red}$. Let $s = [\![E]\!]_{\mathrm{W}}$. Then, by Lemmas 3.2, 3.3, and 3.5, we have (T1-3) from Prop. 3.1. Hence, there exists a weighted trace automaton $\mathcal{A}$ with $\|\mathcal{A}\|_{\mathrm{T}} = \overline{\varphi}(s)$ which equals $[\![E]\!]_{\mathrm{T}}$ by Prop. 2.1. □

4 From automata to expressions

In this section, we want to show that the trace behavior of every weighted trace automaton $\mathcal{A}$ can be described by an expression.

In the following, let $\sqsubseteq$ be some linear order on the alphabet Σ. Let $u \in \Sigma^+$. Then $[u]$ is finite and therefore contains a lexicographically minimal word that we denote $\mathrm{lexNF}(u)$ and call *the lexicographic normal form of* u. Let $\mathrm{LNF}(\Sigma)$ denote the set of words $u \in \Sigma^+$ with $u = \mathrm{lexNF}(u)$.

Lemma 4.1. *Let S be some (possibly non-commutative) semiring, (Σ, I_Σ) an independence alphabet, and $\mathcal{A} = (Q, \Sigma, \lambda, \mu, \gamma)$ some weighted trace automaton over $(\Sigma, \emptyset)$ such that, for any $u, v \in \Sigma^+$ with $u \sim v$, we have $(\|\mathcal{A}\|_{\mathrm{W}}, u) = (\|\mathcal{A}\|_{\mathrm{W}}, v)$. For $u \in \Sigma^+$ let $(s, u) = (\|\mathcal{A}\|_{\mathrm{W}}, u)$ if $u \in \mathrm{LNF}$ and 0 otherwise.*

Then there exist a projection of independence alphabets $\pi : (\Gamma, I_\Gamma) \to (\Sigma, I_\Sigma)$, a homomorphism $c : \Gamma^+ \to (S, \cdot)$, and a language $L \subseteq \Gamma^+$ of words in lexicographic normal form such that (T1-3) hold.

Note that every weighted trace automaton satisfies the above condition on $\|\mathcal{A}\|_{\mathrm{W}}$, but the condition is also satisfied by some weighted automata that are no weighted trace automaton. Hence, this lemma proves that the condition expressed in Prop. 3.1 is also necessary.

Proof. We can assume $\lambda(p), \gamma(p) \in \{0, 1\}$ for all $p \in Q$. Let Γ be the set of transitions of $\mathcal{A}$ and set $((p, a, q), (p', b, q')) \in I_\Gamma$ iff $(a, b) \in I_\Sigma$. The mapping $\pi : (\Gamma, I_\Gamma) \to (\Sigma, I_\Sigma) : (p, a, q) \mapsto a$ is a projection of independence alphabets. The homomorphism c is defined by $c(p, a, q) = \mu(a)_{p,q}$ for all $(p, a, q) \in \Gamma$.

There is some linear order $\sqsubseteq$ on Γ such that $(p, a, q) \sqsubseteq (p', a', q')$ implies $a \sqsubseteq a'$. Then $U \in \mathrm{LNF}$ iff $\pi(U) \in \mathrm{LNF}$ for all $U \in \Gamma^+$. Note that every path in $\mathcal{A}$ is a word over Γ. Then let $L \subseteq \Gamma^+$ be the set of paths in lexicographic normal form from some state $\iota \in \lambda^{-1}(1)$ to some state $f \in \gamma^{-1}(1)$ in $\mathcal{A}$. Since $L \subseteq \mathrm{LNF}$ is regular, [12] implies the regularity of $[L] \subseteq \Gamma^+$, i.e., we showed (T1).

To verify (T2), let $u \in \Sigma^+$. If $u \notin \mathrm{LNF}$, then $\pi^{-1}(u)$ does not contain any word in lexicographic normal form. Thus, in this case, $L \cap \pi^{-1}(u) = \emptyset$ implying that both sides of the equation yield 0. So let $u \in \mathrm{LNF}$. Then $L \cap \pi^{-1}(u)$ equals the set of u-labeled paths from $\lambda^{-1}(1)$ to $\gamma^{-1}(1)$. Hence, (T2) follows from Eq. (1).

As in the verification of (T3) from Section 3.2 (page 402), there is a surjection $f_u : L \cap \pi^{-1}([u]) \to [L] \cap \pi^{-1}(u)$ with $U \sim f_u(U)$ and therefore $c(U) = c(f_u(U))$. In the current setting, this surjection is even injective: If $V, W \in L \cap [\pi^{-1}(u)]$ with $f_u(V) = f_u(W)$, then $V \sim f_u(V) = f_u(W) \sim W$. But then $V, W \in L \subseteq \mathrm{LNF}$ implies $V = W$. Hence f_u is a weight-preserving bijection implying (T3). □

As a consequence, we obtain

Theorem 4.1 (cf. [3, Thm. 1(a)]). *Let S be a semiring and $\mathcal{A}$ a weighted trace automaton over the independence alphabet (Σ, I). Then there exists an mc-rational expression E such that $[\![E]\!]_{\mathrm{T}} = \|\mathcal{A}\|_{\mathrm{T}}$.*

Proof. We can apply Lemma 4.1 since the weighted trace automaton $\mathcal{A}$ satisfies the conditions of that lemma. So let π, Γ etc. be as above such that (T1-3) hold. Since L is regular, there is a regular expression E with $\mathcal{L}(E) = L$. By [4, Lemma 2.1], we can assume that the language $\mathcal{L}(F)$ is mono-alphabetic for every sub-expression F^+ of E. Since $\mathcal{L}(E) \subseteq \mathrm{LNF}$, [12] implies that the language $\mathcal{L}(F)$ is connected for every sub-expression F^+ of the rational expression E. Let the expression G be obtained from E by replacing every appearance of $A \in \Gamma$ with $c(A)\pi(A)$. Then one shows inductively along the construction of the rational expression E:

1. if $([\![G]\!]_{\mathrm{T}}, [u]) \neq 0$, then there exists $U \in \mathcal{L}(E)$ with $\pi(U) \sim u$. Recall that for any sub-expression F^+ of E, the language $\mathcal{L}(F)$ is connected and mono-alphabetic. Hence G is an mc-rational expression.
2. $([\![G]\!]_{\mathrm{W}}, v) = \sum(c(V) \mid V \in \mathcal{L}(E) \cap \pi^{-1}(v))$ which, by (T2) equals $([\![\mathcal{A}]\!]_{\mathrm{W}}, v)$ for $v \in \mathrm{LNF}$ and 0 otherwise.

Then we obtain for $u \in \Sigma^+$:

$$\begin{aligned}([\![G]\!], [u]) &= \sum_{v \in [u]} ([\![G]\!]_{\mathrm{W}}, v) \text{ from Prop. 2.1} \\ &= ([\![G]\!]_{\mathrm{W}}, \mathrm{lexNF}(u)) \\ &= (\|\mathcal{A}\|_{\mathrm{W}}, u) = (\|\mathcal{A}\|_{\mathrm{T}}, [u])\end{aligned}$$

□

5 Discussion

Let S be a commutative semiring. A consequence of Theorems 3.1, 3.2, and 4.1 is the closure of the set of behaviors of weighted trace automata under addition, multiplication, and iteration $^{\mathrm{mc}+}$ (and $^{\mathrm{c}+}$ provided the semiring is idempotent): if $\mathcal{A}_1$ and $\mathcal{A}_2$ are weighted trace automata, by Thm. 4.1, there exist mc-rational expressions E_1 and E_2 with $[\![E_i]\!]_{\mathrm{T}} = \|\mathcal{A}_i\|_{\mathrm{T}}$. Since $E_1 \cdot E_2$ is another mc-rational expression (that is equivalent with some mc-expression by Remark 3.2), its trace behavior $[\![E_1 \cdot E_2]\!]_{\mathrm{T}} = [\![E_1]\!]_{\mathrm{T}} \cdot [\![E_2]\!]_{\mathrm{T}} = \|\mathcal{A}_1\|_{\mathrm{T}} \cdot \|\mathcal{A}_2\|_{\mathrm{T}}$ is the trace behavior of some

weighted trace automaton $\mathcal{A}$ by Thm. 3.2 (similar arguments can be applied for the other operations mentioned above). Since all our proofs (including those referred to from the literature) are effective, the weighted trace automaton $\mathcal{A}$ is computable from $\mathcal{A}_1$ and $\mathcal{A}_2$. Even more explicit automata constructions for these operations were given by Droste & Gastin [3].

References

[1] J. Berstel and C. Reutenauer. *Rational Series and Their Languages.* EATCS Monographs. Springer Verlag, 1988.

[2] J. Berstel and C. Reutenauer. Extension of Brzozowski's derivation calculus of rational expressions to series over the free partially commutative monoids. *Theoretical Computer Science*, 400:144–158, 2008.

[3] M. Droste and P. Gastin. The Kleene-Schützenberger theorem for formal power series in partially commuting variables. *Information and Computation*, 153:47–80, 1999.

[4] M. Droste and D. Kuske. Recognizable languages in divisibility monoids. *Mathematical Structures in Computer Science*, 11:743–770, 2001.

[5] M. Droste, W. Kuich, and H. Vogler, editors. *Handbook of Weighted Automata.* EATCS Monographs in Theoretical Computer Science. Springer, 2009.

[6] V. Diekert and G. Rozenberg. *The Book of Traces.* World Scientific Publ. Co., 1995.

[7] S. Eilenberg. *Automata, Languages and Machines vol. A.* Academic Press, New York, 1974.

[8] S.C. Kleene. Representation of events in nerve nets and finite automata. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, Annals of Mathematics Studies vol. 34, pages 3–40. Princeton University Press, 1956.

[9] W. Kuich and S. Salomaa. *Semirings, Automata, Languages.* Springer Verlag, 1986.

[10] D. Kuske. Schützenberger's theorem on formal power series follows from Kleene's theorem. *Theoretical Computer Science*, 401:243–248, 2008.

[11] A. Mazurkiewicz. Concurrent program schemes and their interpretation. Technical report, DAIMI Report PB-78, Aarhus University, 1977.

[12] E. Ochmański. Regular behaviour of concurrent systems. *Bull. Europ. Assoc. for Theor. Comp. Science*, 27:56–67, 1985.

[13] M.P. Schützenberger. On the definition of a family of automata. *Inf. Control*, 4:245–270, 1961.

[14] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series.* EATCS Texts and Monographs in Computer Science. Springer Verlag, 1978.

Received 6th July 2008

Acta Cybernetica **19** (2009) 411–439.

Recognizable Tree Series with Discounting

Eleni Mandrali and George Rahonis*

Abstract

We consider weighted tree automata with discounting over commutative semirings. For their behaviors we establish a Kleene theorem and an MSO-logic characterization. We introduce also weighted Muller tree automata with discounting over the max-plus and the min-plus semirings, and we show their expressive equivalence with two fragments of weighted MSO-sentences.

Keywords: semirings, discounting, weighted tree automata, rational operations on tree series, weighted Muller tree automata, weighted MSO-logic over finite and infinite trees.

1 Introduction

Weighted tree automata over finite trees have been considered by many researchers (cf. [22] for an extended literature) and have contributed in important areas of Computer Science like code selection [3, 20] and monadic second-order evaluations on graphs [33]. Weighted tree automata models are obtained by classical tree automata, top-down or bottom-up, whose transitions are equipped with weights mainly from a semiring. The weights might model resources used for the execution of transitions, the time needed or reliability. For an excellent survey on weighted tree automata we refer the reader to [22] (cf. also [2]).

If we require that weighted tree automata can work also on infinite trees, then clearly the underlying semiring should admit infinite sums and products satisfying special axioms (cf. [32]). *Discounting* is a common strategy to face problems arising on systems with non-terminating behavior, in particular in economic mathematics, in Markov decision processes, and in game theory (cf. [9, 21, 34]). This method was incorporated for weighted automata over infinite words by Droste and Kuske in [14]. More precisely, the authors considered weighted automata over the max-plus and min-plus semirings, acting on infinite words, and employed a discounting parameter which permitted the summation of infinitely many values. In this way, they achieved a Kleene theorem for the infinitary series obtained as the behaviors of their automata. They also considered weighted automata with discounting

*Department of Mathematics, Aristotle University of Thessaloniki 54124 Thessaloniki, Greece, E-mail: {elemandr,grahonis}@math.auth.gr

over finite words, and they showed a Kleene-Schützenberger theorem for the series accepted by these automata. In [5, 6] further properties of weighted automata with discounting over infinite words were investigated. A weighted MSO-logic with discounting has been introduced in [16] and a Büchi-type characterization of infinitary recognizable series with discounting has been established. Kuich [27] proved Kleene theorems for weighted automata with discounting acting on finite and infinite words over Conway semirings. Recently, in [17] the authors investigated weighted automata with discounting over semirings and finitely generated graded monoids.

In this paper, we introduce weighted tree automata with discounting, acting on finite trees, and our first goal is a Kleene theorem for their behaviors. Furthermore, we consider a weighted MSO-logic on trees with discounting, and in our second main result we show the expressive equivalence of weighted tree automata with discounting with two fragments of this logic. One of these fragments has a purely syntactic definition provided that the underlying semiring is additively locally finite. Our MSO-logic is a slight modification of the MSO-logic in [18] and goes back to the pioneering work of Droste and Gastin [11] (cf. also [13]) in which weighted logics over semirings were considered for the first time. Very recently, in [19] (cf. also [22]) the authors achieved a purely syntactic description in terms of MSO-logic for weighted tree automata over arbitrary semirings. The discounting method for stochastic tree automata has been also considered in [30, 31].

Infinite trees play a crucial role in practical applications, namely in program optimization [23], and in proving termination of non-deterministic or concurrent programs under any reasonable notion of fairness [25]. Furthermore, tree automata over infinite trees contribute to program synthesis in model checking [38]. All these applications are based on the fundamental fact that every program can be described by an infinite tree (cf. [8, 23, 39]). Weighted Muller tree automata were investigated in [32] but for the underlying semirings special completeness axioms were required. Currently, several tools for model checking are built in a weighted setting, in particular over De Morgan algebras (cf. [4, 7, 24]). Therefore, taking into account the contribution of tree automata to program synthesis [38], we wish to study the extension of these models in a weighted setting for semirings, like max-plus and min-plus, which are already used in practical applications. For this we introduce weighted Muller tree automata with discounting, over the max-plus and the min-plus semirings, and in our third main result we state their characterization in terms of weighted (purely syntactically defined) MSO-logic.

The proofs of our results are similar to the corresponding ones in [15, 18, 32]. Nevertheless, they are more technical because of the involvement of discounting parameters. We present only a few of them which are representative for the discounting techniques. The reader can find detailed proofs in [28]. In the paper, we notify the corresponding results from [15, 18, 32] by e.g. (cf. [15], Lm 4.8).

2 Preliminaries

2.1 Trees

We denote by $\mathbb{N}$ the set of natural numbers and let $\mathbb{N}_+ = \mathbb{N}\setminus\{0\}$. The *prefix relation* $\leq$ over $\mathbb{N}^*$ is a partial order defined in the usual way: for every $w, v \in \mathbb{N}^*$, $w \leq v$ iff there exists $u \in \mathbb{N}^*$ such that $wu = v$. A set $A \subseteq \mathbb{N}^*$ is called *prefix-closed* if $v \in A$ implies $w \in A$ for every $w \leq v$.

A *ranked alphabet* Σ is a pair (Σ, rk_Σ) (simply denoted by Σ) where Σ is a finite set and $rk_\Sigma : \Sigma \to \mathbb{N}$. As usual, we set $\Sigma_k = \{\sigma \in \Sigma \mid rk_\Sigma(\sigma) = k\}$ for every $k \geq 0$, and $\deg(\Sigma) = \max\{k \in \mathbb{N} \mid \Sigma_k \neq \emptyset\}$.

A *tree* t *over* Σ is a partial mapping $t : \mathbb{N}_+^* \to \Sigma$ such that the domain $dom(t)$ of t is a non-empty prefix-closed set, and for every $w \in dom(t)$ if $t(w) \in \Sigma_k$ $k \geq 0$, then for $i \in \mathbb{N}_+$, $wi \in dom(t)$ iff $1 \leq i \leq k$. The elements of $dom(t)$ are called the *nodes of* t. For every $\sigma \in \Sigma$ we set $dom_\sigma(t) = \{w \in dom(t) \mid t(w) = \sigma\}$ and, for every $A \subseteq \Sigma$ we let $dom_A(t) = \{w \in dom(t) \mid t(w) \in A\}$. A tree t is called *finite* (resp. *infinite)* if its domain is finite (resp. infinite). As usual, we shall denote by T_Σ (resp. T_Σ^ω) the set of all finite (resp. infinite) trees over Σ. Clearly, $T_\Sigma = \emptyset$ iff $\Sigma_0 = \emptyset$.

The set T_Σ can also be inductively defined as the smallest set T such that (i) $\Sigma_0 \subseteq T$ and (ii) if $k \geq 1, \sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T$, then $\sigma(t_1, \ldots, t_k) \in T$. For every finite set A with $A \cap \Sigma = \emptyset$, we shall write $T_\Sigma(A)$ for the set of finite trees over the ranked alphabet Σ', where $\Sigma'_0 = \Sigma_0 \cup A$ and $\Sigma'_k = \Sigma_k$ for every $k > 0$. The *height* $ht(t)$ of every finite tree $t \in T_\Sigma$ is defined by $ht(t) = \max\{|w| \mid w \in dom(t)\}$ where $|w|$ denotes the length of the word w. Let $a \in \Sigma_0$ and $t \in T_\Sigma$ with $dom_a(t) = \{w_1, \ldots, w_m\}$ such that $w_1 \leq_{lex} \cdots \leq_{lex} w_m$ where $\leq_{lex}$, is the lexicographic order over $\mathbb{N}^*$. Then, for $t_1, \ldots, t_m \in T_\Sigma$ we write $t \cdot_a (t_1, \ldots, t_m)$ for the tree obtained by substituting t_i for a at w_i $(1 \leq i \leq n)$ in t.

> *In the rest of the paper,* Σ *and* Γ *will denote an arbitrary ranked alphabet, if not specified otherwise. Moreover, we assume that* $\Sigma_0 \neq \emptyset$ *and* $\Gamma_0 \neq \emptyset$.

A *relabeling* from Σ to Γ is a surjective mapping $h : \Sigma \to \Gamma$ such that $h(\sigma) \in \Gamma_k$ for every $\sigma \in \Sigma_k$, $k \geq 0$. Then h is extended to a mapping $h : T_\Sigma \to T_\Gamma$ by letting $dom(h(t)) = dom(t)$ and $h(t)(w) = h(t(w))$ for every $t \in T_\Sigma$ and $w \in dom(t)$.

2.2 Semirings

A *semiring* $(K, +, \cdot, 0, 1)$ consists of a set K equipped with two binary operations $+$ and $\cdot$, and two constant elements 0 and 1 such that $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, multiplication distributes over addition, and $0 \cdot a = a \cdot 0 = 0$ for every $a \in K$. If the operations and the constant elements are understood, then the semiring is simply denoted by K. A semiring K is called *commutative* if the monoid $(K, \cdot, 1)$ is commutative. The second main result of our paper will apply to commutative semirings K which are *additively locally finite*, i.e., such

that every finitely generated submonoid of $(K, +, 0)$ is finite. For examples of additively locally finite semirings we refer the reader to [16]. In this paper we mainly deal with the additively locally finite semiring *max-plus* (or *arctic*) $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ where $\mathbb{R}_+ = \{r \in \mathbb{R} \mid r \geq 0\}$ and $-\infty + x = -\infty$ for every $x \in \mathbb{R}_+ \cup \{-\infty\}$. Our results remain valid as well, over the additively locally finite semiring *min-plus* (or *tropical*) $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ with $\infty + x = \infty$ for every $x \in \mathbb{R}_+ \cup \{\infty\}$.

Let K_1 and K_2 be two semirings. A mapping $f : K_1 \to K_2$ is called a *semiring homomorphism* (or simply a *homomorphism*) if $f(a+b) = f(a)+f(b)$ and $f(a \cdot b) = f(a) \cdot f(b)$ for every $a, b \in K_1$, and $f(0) = 0$ and $f(1) = 1$. A homomorphism $f : K \to K$ is an *endomorphism* of K. The set $End(K)$ of all endomorphisms of K is a monoid with operation the usual composition mapping $\circ$ and unit element the identity mapping on K. If no confusion arises, we shall alternatively denote the operation $\cdot$ of K and the composition operation $\circ$ of $End(K)$ by concatenation.

Example 1. Consider the max-plus semiring $\mathbb{R}_{\max}$ and extend the multiplication $\cdot$ over $\mathbb{R}_+$ by letting $p \cdot (-\infty) = (-\infty) \cdot p = -\infty$ for every $p \in \mathbb{R}_+ \cup \{-\infty\}$. Then the mapping $\overline{p} : \mathbb{R}_{\max} \to \mathbb{R}_{\max}$ $(p \in \mathbb{R}_+)$ given by $x \longmapsto p \cdot x$ is an endomorphism of $\mathbb{R}_{\max}$. Conversely, every endomorphism of $\mathbb{R}_{\max}$ is of this form (cf. [14], Lm. 15).

In the rest of the paper, K will denote an arbitrary commutative semiring if not specified otherwise.

2.3 Discounting

A *discounting over Σ and K* is a family $\Phi = (\Phi_k)_{k \geq 1}$ of mappings $\Phi_k : \Sigma_k \to (End(K))^k$ for $k \geq 1$. For every $\sigma \in \Sigma_k$ $(k \geq 1)$ we shall write $(\Phi^1_\sigma, \ldots, \Phi^k_\sigma)$ for the k-tuple $\Phi_k(\sigma)$. If no confusion arises with the rank of σ, then we simply denote $\Phi_k(\sigma)$ by Φ_σ. The discounting Φ is alternatively called a *Φ-discounting*. For every $t \in T_\Sigma$ and every $w \in dom(t)$, we define the endomorphism Φ^t_w of K as follows:

$$\Phi^t_w = \begin{cases} id & \text{if } w = \varepsilon \\ \Phi^{i_1}_{t(\varepsilon)} \circ \Phi^{i_2}_{t(i_1)} \circ \ldots \circ \Phi^{i_n}_{t(i_1 \ldots i_{n-1})} & \text{if } w = i_1 \ldots i_n,\ i_1, \ldots, i_n \in \mathbb{N}_+, n > 0 \end{cases}$$

where id is the identity endomorphism of K.

In Sections 3-5, Φ will denote a discounting over Σ and K.

2.4 Tree series

A *formal tree series* (or *tree series* for short) *over Σ and K*, is a mapping $S : T_\Sigma \to K$. As usual we denote by (S, t) the *coefficient* $S(t)$ for every $t \in T_\Sigma$. The *support* of S is the tree language $supp(S) = \{t \in T_\Sigma \mid (S, t) \neq 0\}$. The class of all tree series over Σ and K is denoted by $K\langle\langle T_\Sigma \rangle\rangle$, and the class of *polynomials* (i.e., tree series with finite support) is denoted by $K\langle T_\Sigma \rangle$.

For every tree language $L \subseteq T_\Sigma$, the *characteristic series* $1_L \in K\langle\langle T_\Sigma\rangle\rangle$ *of* L *with respect to* K is determined by $(1_L, t) = 1$ if $t \in L$, and $(1_L, t) = 0$ otherwise, for every $t \in T_\Sigma$. Let $S, T \in K\langle\langle T_\Sigma\rangle\rangle$ and $k \in K$. The sum $S+T$, the scalar products kS and Sk, and the Hadamard product $S \odot T$ are defined by $(S+T, t) = (S, t) + (T, t)$, $(kS, t) = k \cdot (S, t)$ and $(Sk, t) = (S, t) \cdot k$, and $(S \odot T, t) = (S, t) \cdot (T, t)$ for every $t \in T_\Sigma$. Clearly, $(K\langle\langle T_\Sigma\rangle\rangle, +, \odot, \mathbf{0}, \mathbf{1})$ and $(K\langle T_\Sigma\rangle, +, \odot, \mathbf{0}, \mathbf{1})$ are commutative semirings, where $\mathbf{0}$ is the tree series (over Σ and K) with all its coefficients being 0, and $\mathbf{1}$ is the tree series (over Σ and K) with all its coefficients being 1.

Let $h : \Sigma \to \Gamma$ be a relabeling. For every tree series $S \in K\langle\langle T_\Sigma\rangle\rangle$ the tree series $h(S) \in K\langle\langle T_\Gamma\rangle\rangle$ is defined for every $s \in T_\Gamma$ by $(h(S), s) = \sum_{t \in h^{-1}(s)} (S, t)$. Similarly, for every $T \in K\langle\langle T_\Gamma\rangle\rangle$ the tree series $h^{-1}(T) \in K\langle\langle T_\Sigma\rangle\rangle$ is determined by $(h^{-1}(T), t) = (T, h(t))$ for every $t \in T_\Sigma$.

3 Φ-recognizable tree series

In this section, we study Φ-recognizable tree series obtained as behaviors of weighted tree automata with Φ-discounting. Intuitively, for every input tree t the weight of every node of t is discounted according to the distance of the node from the root of t; the longer the distance is the greater the grade of discounting is; nodes of the same level get a weight with the same grade of discounting. Our weighted tree automata are bottom-up models without initial distribution. By standard automata constructions, it can be seen that they are equivalent to weighted tree automata with initial distribution. Furthermore, they are equivalent to the corresponding top-down models, with and without terminal distribution (cf. [28]). Firstly, we introduce our weighted tree automata with Φ-discounting and we state normalization results. Then, in Subsection 3.2 we investigate closure properties of Φ-recognizable tree series.

3.1 Weighted tree automata with Φ-discounting

Definition 1. *A* weighted tree automaton with Φ-discounting *(*Φ-wta *for short)* over Σ and K *is a triple* $\mathcal{M} = (Q, wt, ter)$ *where* Q *is the* finite state set, $wt : \bigcup_{k \geq 0} Q^k \times \Sigma_k \times Q \to K$ *is the* mapping assigning weights to the transitions *of the automaton, and* $ter : Q \to K$ *is* the final distribution.

Let $t \in T_\Sigma(Q)$ (without any loss we assume that $\Sigma \cap Q = \emptyset$) and $P \subseteq Q$. A *run of* $\mathcal{M}$ *over* t *using* P is a mapping $r_t : dom(t) \to Q$ such that $r_t(w) = t(w)$ for every $w \in dom_Q(t)$ and $r_t(w) \in P$ for every $w \in dom(t) \setminus (dom_Q(t) \cup \{\varepsilon\})$. The run r_t is called a *q-run* whenever $r_t(\varepsilon) = q$. We shall denote by $R^P_{\mathcal{M}}(t, q)$ the set of all q-runs of $\mathcal{M}$ over t using P, and by $R_{\mathcal{M}}(t, q)$ the set $R^Q_{\mathcal{M}}(t, q)$. Moreover, we let $R_{\mathcal{M}}(t) = \bigcup_{q \in Q} R_{\mathcal{M}}(t, q)$.

The *weight of a run* $r_t \in R^P_{\mathcal{M}}(t,q)$ *at* $w \in dom(t)$ is given by

$$wt(r_t,w) = \begin{cases} wt((r_t(w1),\ldots,r_t(w.rk_\Sigma(t(w)))),t(w),r_t(w)) & \text{if } t(w)\in\Sigma_k, k\geq 0 \\ 1 & \text{if } t(w)\in Q. \end{cases}$$

The *running* Φ*-weight of* r_t, denoted by $rweight_{\mathcal{M}}(r_t)$ (or simply $rweight(r_t)$), is the value

$$rweight_{\mathcal{M}}(r_t) = \prod_{w\in dom(t)} \Phi^t_w(wt(r_t,w))$$

and the Φ*-weight of* r_t, denoted by $weight_{\mathcal{M}}(r_t)$ (or simply $weight(r_t)$), is given by

$$weight_{\mathcal{M}}(r_t) = rweight_{\mathcal{M}}(r_t)\cdot ter(r_t(\varepsilon)).$$

For every $P\subseteq Q, q\in Q$ we let $\|\mathcal{M}\|(P,q)$ be the tree series in $K\langle\langle T_\Sigma(Q)\rangle\rangle$ determined by

$$(\|\mathcal{M}\|(P,q),t) = \begin{cases} \sum\limits_{r_t\in R^P_{\mathcal{M}}(t,q)} rweight_{\mathcal{M}}(r_t) & \text{if } t\in T_\Sigma(Q)\setminus Q \\ 0 & \text{otherwise.} \end{cases}$$

The Φ*-behavior* (or simply *behavior*) of $\mathcal{M}$ is the tree series $\|\mathcal{M}\|\in K\langle\langle T_\Sigma\rangle\rangle$ defined for every $t\in T_\Sigma$ by

$$(\|\mathcal{M}\|,t) = \sum_{r_t\in R_{\mathcal{M}}(t)} weight_{\mathcal{M}}(r_t).$$

Clearly,

$$(\|\mathcal{M}\|,t) = \sum_{q\in Q}(\|\mathcal{M}\|(Q,q),t)\cdot ter(q).$$

for every $t\in T_\Sigma$

A tree series $S\in K\langle\langle T_\Sigma\rangle\rangle$ is called Φ*-recognizable* if there is a Φ-wta $\mathcal{M}$ over Σ and K such that $S=\|\mathcal{M}\|$. We shall denote by $Rec(\Sigma,K,\Phi)$ the class of all Φ-recognizable tree series over Σ and K. Clearly, if our Φ-discounting employs only the identity mapping on K, i.e, $\Phi_\sigma=(id,\ldots,id)$ for every $\sigma\in\Sigma_k$ $(k\geq 1)$, then $Rec(\Sigma,K,\Phi)=Rec(\Sigma,K)$ the class of recognizable formal tree series over Σ and K (cf. [1, 22]). Two Φ-wta $\mathcal{M}$ and $\mathcal{M}'$ are *equivalent* if $\|\mathcal{M}\|=\|\mathcal{M}'\|$.

Next, we give examples of Φ-recognizable tree series over $\mathbb{R}_{\max}$ obtained as behaviors of deterministic Φ-wta. More precisely, a (Φ-)wta $\mathcal{M}=(Q,wt,ter)$ over Σ and K is called *deterministic* (cf. [2]) if for every $k\geq 0,\sigma\in\Sigma_k$, and $q_1,\ldots,q_k\in Q$ there is at most one $q\in Q$ such that $wt((q_1,\ldots,q_k),\sigma,q)\neq 0$.

Example 2. Let Σ be a ranked alphabet with $\Sigma_0=\{a\},\Sigma_2=\{\sigma,\gamma\}$, and $\Sigma_3=\{\delta\}$. We consider the Φ-wta $\mathcal{M}=(\{q\},wt,ter)$ over Σ and $\mathbb{R}_{\max}$ with its weight assignment mapping defined by $wt((q,q),\sigma,q)=1$ and $wt(a,q)=wt((q,q),\gamma,q)=$

$wt((q,q,q),\delta,q) = 0$. The final distribution is given by $ter(q) = 0$. We define a Φ-discounting over Σ and $\mathbb{R}_{\max}$ specified by $\Phi_\sigma = (\overline{1},\overline{1})$, $\Phi_\gamma = (\overline{0},\overline{0})$, and $\Phi_\delta = (\overline{0},\overline{0},\overline{0})$ (cf. Example 1).

Then for every $t \in T_\Sigma$ the coefficient $(\|\mathcal{M}\|, t)$ equals the number of occurrences of σ in the greatest initial σ-subtree of t. One can easily show that there is no deterministic wta without discounting over Σ and $\mathbb{R}_{\max}$ accepting the same tree series.

Example 3. Consider the ranked alphabet Σ of the previous example and let $t \in T_\Sigma$. We say that the *pattern* $\delta(\sigma,\sigma,\delta)$ occurs in t if there are trees $t', s, s_i \in T_\Sigma$ $(1 \leq i \leq 7)$ such that $t = t' \cdot_a s$ and $s = \delta(\sigma(s_1,s_2),\sigma(s_3,s_4),\delta(s_5,s_6,s_7))$. We construct a deterministic Φ-wta $\mathcal{M} = (Q, wt, ter)$ over Σ and $\mathbb{R}_{\max}$, whose Φ-behavior returns for every input tree $t \in T_\Sigma$ the number of occurrences of the pattern $\delta(\sigma,\sigma,\delta)$ in the greatest initial subtree of t which does not contain any symbol γ. Our Φ-discounting now is given by $\Phi_\sigma = (\overline{1},\overline{1})$, $\Phi_\gamma = (\overline{0},\overline{0})$, and $\Phi_\delta = (\overline{1},\overline{1},\overline{1})$. The Φ-wta $\mathcal{M}$ is determined by $Q = \{q_1,q_2,q_3\}$, $ter(q) = 0$ for every $q \in Q$, and

- $wt(a,q_1) = 0$
- $wt((p_1,p_2),\sigma,q_2) = wt((p_1,p_2),\gamma,q_1) = 0$ for every $p_1,p_2 \in Q$,
- $wt((q_2,q_2,q_3),\delta,q_3) = 1$, and
- $wt((p_1,p_2,p_3),\delta,q_3) = 0$ for every $p_1,p_2,p_3 \in Q$ with $(p_1,p_2,p_3) \neq (q_2,q_2,q_3)$.

Any other transition is assigned the value $-\infty$. Clearly, $\mathcal{M}$ is deterministic and by standard arguments we can show that $\|\mathcal{M}\|$ cannot be accepted by any deterministic wta without discounting over Σ and $\mathbb{R}_{\max}$.

Next we establish two normalized forms of Φ-wta; they will be used for the proofs of the results in Section 4.

A Φ-wta $\mathcal{M} = (Q, wt, ter)$ is *final weight normalized* (cf. [15], Def. 4.7) if there is one state $q_f \in Q$ such that

- $ter(q_f) = 1$ and, for every $q \in Q$ with $q \neq q_f$, $ter(q) = 0$,
- for every $k > 0, \sigma \in \Sigma_k, q_1,\ldots,q_k,q \in Q$, if there is an $1 \leq i \leq k$ with $q_i = q_f$, then $wt((q_1,\ldots,q_k),\sigma,q) = 0$.

In this case we write $\mathcal{M} = (Q, wt, q_f)$.

Lemma 1. *(cf. [15], Lm 4.8) For every Φ-wta $\mathcal{M}$ there is an equivalent final weight normalized Φ-wta $\mathcal{M}'$. Moreover, $\mathcal{M}'$ can be chosen to have one more state than $\mathcal{M}$.*

Let $a \in \Sigma_0$. A tree series $S \in K\langle\langle T_\Sigma\rangle\rangle$ is called *a-proper* if $(S,a) = 0$. We shall denote by $K^a\langle\langle T_\Sigma\rangle\rangle$ the class of all a-proper tree series over Σ and K. Consider a Φ-wta $\mathcal{M} = (Q, wt, ter)$ over Σ and K. We let $I_a = \{q \in Q \mid wt(a,q) \neq 0\}$, and we call I_a the set of *initial a-states* of $\mathcal{M}$. The Φ-wta $\mathcal{M}$ is called *initial a-state normalized* (cf. [15], Def. 4.10) if there is a state $q_a \in Q$ such that $I_a = \{q_a\}$, $wt(a,q_a) = 1$, and $wt((q_1,\ldots,q_k),\sigma,q_a) = 0$ for every $\sigma \in \Sigma \setminus \{a\}$.

Lemma 2. *(cf. [15], Lm. 4.11) Let $\mathcal{M}=(Q,wt,ter)$ be a Φ-wta over Σ and K and $a \in \Sigma_0$. Then there is an initial a-state normalized Φ-wta $\mathcal{M}'$ such that $(\|\mathcal{M}'\|,t) = (\|\mathcal{M}\|,t)$ for every $t \in T_\Sigma \setminus \{a\}$. Moreover, $\mathcal{M}'$ can be chosen to have one more state than $\mathcal{M}$.*

Proof. We construct the Φ-wta $\mathcal{M}' = (Q', wt', ter')$ with $Q' = Q \cup \{q_a\}$. The final distribution ter' is given by $ter'(q_a) = 0$ and by $ter'(q) = ter(q)$ for every $q \in Q$. The weight assignment mapping wt' is defined as follows:

- $wt'((q_1,\ldots,q_k),\sigma,q) =$

$$\sum \left\{ \left(\prod_{\substack{1\le i\le k \\ q_i=q_a}} \Phi^i_\sigma(wt(a,p_i)) \right) \cdot wt((p_1,\ldots,p_k),\sigma,q) \mid p_1,\ldots,p_k \in Q, \text{ and } p_i = q_i \text{ if } q_i \in Q \right\}$$

  for every $k \ge 0, \sigma \in \Sigma_k \setminus \{a\}, q_1,\ldots,q_k \in Q'$, and $q \in Q$

- $wt'((q_1,\ldots,q_k),\sigma,q_a) = 0$ for every $k \ge 0, \sigma \in \Sigma_k \setminus \{a\}$, and $q_1,\ldots,q_k \in Q'$

- $wt'(a,q) = 0$ for every $q \in Q$

- $wt'(a,q_a) = 1$.

Obviously, $\mathcal{M}'$ is initial a-state normalized and q_a is the initial a-state. Observe that for every $t \in T_\Sigma$, $r'_t \in R_{\mathcal{M}'}(t)$, and $w \in dom(t)$, if $t(w) \neq a$ and $r'_t(w) = q_a$, then $weight_{\mathcal{M}'}(r'_t) = 0$. We will show that $(\|\mathcal{M}'\|,t) = (\|\mathcal{M}\|,t)$ for every $t \in T_\Sigma \setminus \{a\}$. Let $t \neq a$. For every $q \in Q$, we define the mapping $v : R_{\mathcal{M}}(t,q) \to R_{\mathcal{M}'}(t,q)$ as follows. For every run $r_t \in R_{\mathcal{M}}(t,q)$ and every $w \in dom(t)$ we put

$$(v(r_t))(w) = \begin{cases} r_t(w) & \text{if } t(w) \neq a \\ q_a & \text{otherwise.} \end{cases}$$

We set $pre_a(t) = \{w \in dom(t) \mid \text{there exists an } i \in \mathbb{N}_+ \text{ such that } t(wi) = a\}$, i.e., the set of all nodes of t which are predecessors of the a-labeled nodes. Let $pre_a(t) = \{w_1,\ldots,w_m\}$. Then for every $1 \le j \le m$, we set $dom_{a,j}(t) = \{i \mid t(w_j i) = a\} = \{i_{j1},\ldots,i_{jk_j}\}$ (with $i_{j1} < \ldots < i_{jk_j}$) which indicates the set of all a-labeled nodes following w_j. Clearly, $dom_a(t) = \bigcup_{1\le j\le m} \{w_j i \mid i \in dom_{a,j}(t)\}$. Finally we define the *a-surrounding* of t to be the set $sur_a(t) = pre_a(t) \cup dom_a(t)$. Let $q \in Q$ and $r'_t \in R_{\mathcal{M}'}(t,q)$ with $r'_t(w) = q_a$ for every $w \in dom_a(t)$. Then we

calculate

$$\sum_{r_t \in v^{-1}(r'_t) w \in sur_a(t)} \prod \Phi^t_w \left(wt\left(r_t, w\right)\right)$$

$$= \sum_{r_t \in v^{-1}(r'_t) 1 \le j \le m} \prod \left(\prod_{i \in dom_{a,j}(t)} \Phi^t_{w_j i} \left(wt\left(r_t, w_j i\right)\right) \right) \cdot \Phi^t_{w_j} \left(wt\left(r_t, w_j\right)\right)$$

$$= \sum_{r_t \in v^{-1}(r'_t) 1 \le j \le m} \prod \left(\prod_{1 \le l \le k_j} \Phi^t_{w_j i_{jl}} \left(wt\left(a, r_t\left(w_j i_{jl}\right)\right)\right) \right)$$

$$\cdot \Phi^t_{w_j} \left(wt \left(\begin{pmatrix} r_t\left(w_j 1\right), \ldots, r_t\left(w_j\left(i_{j1}-1\right)\right), r_t\left(w_j i_{j1}\right), \\ r_t\left(w_j\left(i_{j1}+1\right)\right), \ldots, r_t\left(w_j\left(i_{j2}-1\right)\right), r_t\left(w_j i_{j2}\right), \\ \ldots, r_t\left(w_j\left(i_{j(k_j-1)}+1\right)\right), \ldots, r_t\left(w_j\left(i_{jk_j}-1\right)\right), \\ r_t\left(w_j i_{jk_j}\right), r_t\left(w_j\left(i_{jk_j}+1\right)\right), \ldots, r_t\left(w_j \rho_j\right) \\ t(w_j), r_t\left(w_j\right) \end{pmatrix} \right) \right)$$

$$= \prod_{1 \le j \le m} \sum \left\{ \Phi^t_{w_j} \left(wt \left(\begin{pmatrix} \left(\prod_{1 \le l \le k_j} \Phi^t_{w_j i_{jl}} \left(wt\left(a, p_{jl}\right)\right) \right) \cdot \\ r'_t\left(w_j 1\right), \ldots, r'_t\left(w_j\left(i_{j1}-1\right)\right), p_{j1}, \\ r'_t\left(w_j\left(i_{j1}+1\right)\right), \ldots, r'_t\left(w_j\left(i_{j2}-1\right)\right), \\ p_{j2}, \ldots, p_{jk_j}, \ldots, r'_t\left(w_j \rho_j\right) \\ t(w_j), r'_t\left(w_j\right) \\ \mid p_{j1}, \ldots, p_{jk_j} \in Q \end{pmatrix} \right) \right) \right\}$$

where for every $1 \le j \le m$ we assume that $rk(t\left(w_j\right)) = \rho_j$. On the other side

$$\prod_{w \in sur_a(t)} \Phi^t_w \left(wt'\left(r'_t, w\right)\right)$$

$$= \prod_{1 \le j \le m} \left(\Phi^t_{w_j} \left(wt'\left(r'_t, w_j\right)\right) \cdot \prod_{i \in dom_{a,j}(t)} \Phi^t_{w_j i} \left(wt'\left(r'_t, w_j i\right)\right) \right)$$

$$= \prod_{1 \le j \le m} \Phi^t_{w_j} \left(wt'\left(r'_t, w_j\right)\right)$$

$$= \prod_{1 \le j \le m} \Phi^t_{w_j} \left(wt'\left(\left(r'_t\left(w_j 1\right), \ldots, r'_t\left(w_j \rho_j\right)\right), t(w_j), r'_t\left(w_j\right)\right)\right),$$

which equals

$$\prod_{1\le j\le m}\Phi^{t}_{w_j}\left(\sum\left\{\cdot wt\left(\begin{pmatrix}\prod_{1\le l\le k_j}\Phi^{i_{jl}}_{t(w_j)}\left(wt\left(a,p_{jl}\right)\right)\end{pmatrix}\cdot\left(\begin{array}{c}r'_t\left(w_j1\right),\ldots,r'_t\left(w_j\left(i_{j1}-1\right)\right),\\ p_{j1},r'_t\left(w_j\left(i_{j1}+1\right)\right),\ldots,\\ r'_t\left(w_j\left(i_{j2}-1\right)\right),p_{j2},\ldots,p_{jk_j},\ldots,\\ r'_t\left(w_j\rho_j\right)\\ t\left(w_j\right),r'_t\left(w_j\right)\\ \mid p_{j1},\ldots,p_{jk_j}\in Q\end{array}\right)\right\}\right)$$

$$=\prod_{1\le j\le m}\sum\left\{\Phi^{t}_{w_j}\left(wt\left(\left(\begin{array}{c}r'_t\left(w_j1\right),\ldots,r'_t\left(w_j\left(i_{j1}-1\right)\right),\\ p_{j1},r'_t\left(w_j\left(i_{j1}+1\right)\right),\ldots,\\ r'_t\left(w_j\left(i_{j2}-1\right)\right),p_{j2},\ldots,p_{jk_j},\\ \ldots,r'_t\left(w_j\rho_j\right)\\ t(w_j),r'_t\left(w_j\right)\\ \mid p_{j1},\ldots,p_{jk_j}\in Q\end{array}\right)\right)\cdot\left(\prod_{1\le l\le k_j}\Phi^{t}_{w_ji_{j_l}}\left(wt\left(a,p_{jl}\right)\right)\right)\right\}.$$

Now, we can easily show that $(\|\mathcal{M}\|,t)=(\|\mathcal{M}'\|,t)$. □

3.2 Properties of Φ-recognizable tree series

Proposition 1. *(i) (cf. [18], Lm. 3.3) The class $Rec(\Sigma,K,\Phi)$ is closed under sum, scalar product, and Hadamard product.*

(ii) (cf. [18], Lm. 3.4) Let $h:\Sigma\to\Gamma$ be a relabeling. Furthermore, for the Φ-discounting over Σ and K assume that $\Phi_\sigma=\Phi_{\sigma'}$ whenever $h(\sigma)=h(\sigma')$ for every $\sigma,\sigma'\in\Sigma_k, k\ge 1$. Let $\Phi'=(\Phi'_k)_{k\ge1}$ be the discounting over Γ and K determined for every $\gamma\in\Gamma_k$ $(k\ge1)$ by $\Phi'_\gamma=\Phi_\sigma$ for every $\sigma\in\Sigma_k$ $(k\ge1)$ with $h(\sigma)=\gamma$. If $S\in Rec(\Sigma,K,\Phi)$, then $h(S)\in Rec(\Gamma,K,\Phi')$. Furthermore, if $T\in Rec(\Gamma,K,\Phi')$, then $h^{-1}(T)\in Rec(\Sigma,K,\Phi)$.[1]

(iii) (cf. [18], Lm. 3.3) Let $L\subseteq T_\Sigma$ be a recognizable tree language. Then $1_L\in Rec(\Sigma,K,\Phi)$.

A tree series $S\in K\langle\langle T_\Sigma\rangle\rangle$ is called a *recognizable step function* if $S=\sum_{1\le j\le n}k_j1_{L_j}$ where $k_j\in K$ and $L_j\subseteq T_\Sigma$ $(1\le j\le n$ and $n\in\mathbb{N})$ are recognizable tree languages. By Proposition 1 such a tree series is Φ-recognizable. The class of recognizable tree languages is closed under the Boolean operations, therefore for every recognizable step function $S=\sum_{1\le j\le n}k_j1_{L_j}$ we may assume the family $(L_j)_{j\in J}$ to be a partition of T_Σ.

[1] Statement (ii) requires that $\deg(\Sigma)=\deg(\Gamma)$ which is guaranteed by the surjectivity of the relabeling h (cf. definition of relabeling on page 413).

Proposition 2. *(i) (cf. [12]) The class of all recognizable step functions over Σ and K is closed under sum, scalar product, and Hadamard product.*

(ii) Let $h : \Sigma \to \Gamma$ be a relabeling. If $T \in K\langle\langle T_\Gamma \rangle\rangle$ is a recognizable step function, then $h^{-1}(T) \in K\langle\langle T_\Sigma \rangle\rangle$ is also a recognizable step function.

(iii) (cf. [16], Prop. 16) Let K be additively locally finite, $h : \Sigma \to \Gamma$ a relabeling, and $S \in K\langle\langle T_\Sigma \rangle\rangle$ a recognizable step function. Then the tree series $h(S) \in K\langle\langle T_\Gamma \rangle\rangle$ is also a recognizable step function.

4 Φ-rational tree series and a Kleene theorem

In this section we introduce the Φ-rational operations on formal tree series and we show a Kleene theorem for Φ-recognizable tree series.

Let $k \geq 1, \sigma \in \Sigma_k$. The *Φ-top-concatenation with σ* is the operation $\sigma_\Phi : K\langle\langle T_\Sigma \rangle\rangle^k \to K\langle\langle T_\Sigma \rangle\rangle$ on tree series defined for every $S_1, \ldots, S_k \in K\langle\langle T_\Sigma \rangle\rangle$ and $t \in T_\Sigma$ by

$$\left(\sigma_\Phi\left(S_1, \ldots, S_k\right), t\right) = \begin{cases} \Phi_\sigma^1\left(\left(S_1, t_1\right)\right) \cdot \ldots \cdot \Phi_\sigma^k\left(\left(S_k, t_k\right)\right) & \text{if } t = \sigma\left(t_1, \ldots, t_k\right) \\ 0 & \text{otherwise.} \end{cases}$$

Let $S, T \in K\langle\langle T_\Sigma \rangle\rangle$ and $a \in \Sigma_0$. The *(a, Φ)-concatenation of S and T* is the tree series $S \cdot_{a,\Phi} T \in K\langle\langle T_\Sigma \rangle\rangle$ defined for every $t \in T_\Sigma$ by

$$\left(S \cdot_{a,\Phi} T, t\right) = \sum_{\substack{s, t_1, \ldots, t_r \in T_\Sigma, t = s \cdot_a (t_1, \ldots, t_r) \\ dom_a(s) = \{w_1, \ldots, w_r\}}} (S, s) \cdot \Phi_{w_1}^t\left(\left(T, t_1\right)\right) \cdot \ldots \cdot \Phi_{w_r}^t\left(\left(T, t_r\right)\right).$$

Proposition 3. *(cf. [15], Lm. 3.3) The (a, Φ)-concatenation of tree series is associative, i.e., for every $S, T, R \in K\langle\langle T_\Sigma \rangle\rangle$ it holds $S \cdot_{a,\Phi} (T \cdot_{a,\Phi} R) = (S \cdot_{a,\Phi} T) \cdot_{a,\Phi} R$.*

Proof. For every $t \in T_\Sigma$ we have

$$\begin{aligned}
&\left(S \cdot_{a,\Phi} \left(T \cdot_{a,\Phi} R\right), t\right) \\
&= \sum_{\substack{s, t_1, \ldots, t_r \in T_\Sigma, t = s \cdot_a (t_1, \ldots, t_r) \\ dom_a(s) = \{w_1, \ldots, w_r\}}} (S, s) \cdot \prod_{i=1}^{r} \Phi_{w_i}^t\left(\left(T \cdot_{a,\Phi} R, t_i\right)\right) \\
&= \sum_{\substack{s, t_1, \ldots, t_r \in T_\Sigma, t = s \cdot_a (t_1, \ldots, t_r) \\ dom_a(s) = \{w_1, \ldots, w_r\}}} (S, s) \\
&\quad \cdot \prod_{i=1}^{r} \Phi_{w_i}^t \left(\sum_{\substack{v_i, u_{i1}, \ldots, u_{in_i} \in T_\Sigma, t_i = v_i \cdot_a (u_{i1}, \ldots, u_{in_i}) \\ dom_a(v_i) = \{w_1^i, \ldots, w_{n_i}^i\}}} (T, v_i) \cdot \prod_{j_i=1}^{n_i} \Phi_{w_{j_i}^i}^{t_i}\left(\left(R, u_{ij_i}\right)\right) \right)
\end{aligned}$$

$$= \sum_{\substack{s,t_1,\ldots,t_r\in T_\Sigma, t=s\cdot_a(t_1,\ldots,t_r)\\ dom_a(s)=\{w_1,\ldots,w_r\}}} (S,s)$$
$$\cdot \prod_{i=1}^{r} \left(\sum_{\substack{v_i,u_{i1},\ldots,u_{in_i}\in T_\Sigma, t_i=v_i\cdot_a(u_{i1},\ldots,u_{in_i})\\ dom_a(v_i)=\{w_1^i,\ldots,w_{n_i}^i\}}} \Phi^t_{w_i}((T,v_i)) \cdot \prod_{j_i=1}^{n_i} \Phi^t_{w_i}\circ\Phi^{t_i}_{w^i_{j_i}}((R,u_{ij_i})) \right)$$

$$= \sum_{\substack{s,t_1,\ldots,t_r,v_i,u_{i1},\ldots,u_{in_i}\in T_\Sigma\\ t=s\cdot_a(t_1,\ldots,t_r), t_i=v_i\cdot_a(u_{i1},\ldots,u_{in_i})\\ dom_a(s)=\{w_1,\ldots,w_r\}, dom_a(v_i)=\{w_1^i,\ldots,w_{n_i}^i\}}} (S,s)$$
$$\cdot \prod_{i=1}^{r} \left(\Phi^t_{w_i}((T,v_i)) \cdot \prod_{j_i=1}^{n_i} \Phi^t_{w_i w^i_{j_i}}((R,u_{ij_i})) \right).$$

On the other side

$$((S\cdot_{a,\Phi} T)\cdot_{a,\Phi} R,t)$$
$$= \sum_{\substack{v,u_1,\ldots,u_q\in T_\Sigma, t=v\cdot_a(u_1,\ldots,u_q)\\ dom_a(v)=\{w_1,\ldots,w_q\}}} \left((S\cdot_{a,\Phi} T,v)\cdot \prod_{j=1}^{q} \Phi^t_{w_j}((R,u_j)) \right)$$

$$= \sum_{\substack{v,u_1,\ldots,u_q\in T_\Sigma, t=v\cdot_a(u_1,\ldots,u_q)\\ dom_a(v)=\{w_1,\ldots,w_q\}}} \left(\sum_{\substack{s,s_1,\ldots,s_r\in T_\Sigma, v=s\cdot_a(s_1,\ldots,s_r)\\ dom_a(s)=\{w'_1,\ldots,w'_r\}}} (S,s)\cdot \prod_{i=1}^{r} \Phi^v_{w'_i}((T,s_i)) \cdot \prod_{j=1}^{q} \Phi^t_{w_j}((R,u_j)) \right)$$

$$= \sum_{\substack{v,u_1,\ldots,u_q,s,s_1,\ldots,s_r\in T_\Sigma\\ t=v\cdot_a(u_1,\ldots,u_q), v=s\cdot_a(s_1,\ldots,s_r)\\ dom_a(v)=\{w_1,\ldots,w_q\}, dom_a(s)=\{w'_1,\ldots,w'_r\}}} (S,s)\cdot \prod_{i=1}^{r} \Phi^t_{w'_i}((T,s_i)) \cdot \prod_{j=1}^{q} \Phi^t_{w_j}((R,u_j)).$$

The last equality is true since every node of v is also a node of t. Clearly, there is a one to one correspondence between the two ways of decomposing t. This also implies that the occurred endomorphisms at each node of the corresponding decompositions coincide. Therefore, we get $(S\cdot_{a,\Phi}(T\cdot_{a,\Phi} R),t) = ((S\cdot_{a,\Phi} T)\cdot_{a,\Phi} R,t)$ for every $t\in T_\Sigma$ and thus $S\cdot_{a,\Phi}(T\cdot_{a,\Phi} R) = (S\cdot_{a,\Phi} T)\cdot_{a,\Phi} R$. □

Following [15] we introduce power discounted iterations of tree series. More precisely, let $S \in K\langle\langle T_\Sigma \rangle\rangle$ and $a \in \Sigma_0$. The *nth* (a,Φ)-*iteration of* S is the tree series $S^n_{a,\Phi} \in K\langle\langle T_\Sigma \rangle\rangle$ defined inductively as follows:

(i) $S^0_{a,\Phi} = \mathbf{0}$ and

(ii) $S^{n+1}_{a,\Phi} = S \cdot_{a,\Phi} S^n_{a,\Phi} + 1a$ for every $n \geq 0$.

Lemma 3. *(cf. [15], Lm. 3.10) Let* $S \in K^a\langle\langle T_\Sigma \rangle\rangle$ *and* $t \in T_\Sigma$. *If* $n \geq ht(t)+1$, *then* $\left(S^{n+1}_{a,\Phi}, t\right) = \left(S^n_{a,\Phi}, t\right)$.

Now we are ready to define the (a,Φ)-Kleene star of a-proper tree series.

Definition 2. *(cf. [15], Def. 3.11) Let* $S \in K^a\langle\langle T_\Sigma \rangle\rangle$. *The* (a,Φ)-Kleene star *(or simply* (a,Φ)-star*)* of S *is a tree series* $S^*_{a,\Phi} \in K\langle\langle T_\Sigma \rangle\rangle$ *which is defined in the following way. For every* $t \in T_\Sigma$ *we set* $\left(S^*_{a,\Phi}, t\right) = \left(S^{ht(t)+1}_{a,\Phi}, t\right)$.

Lemma 4. *(cf. [15], Lm. 3.13) Let* $S \in K^a\langle\langle T_\Sigma \rangle\rangle$. *Then* $S^*_{a,\Phi} = S \cdot_{a,\Phi} S^*_{a,\Phi} + 1a$.

Definition 3. *The set Rat-Exp*(Σ, K, Φ) *of* Φ-rational expressions *(over* Σ *and* K*) is defined inductively as the smallest set* R *satisfying the following conditions. For every* Φ*-rational expression* $\zeta \in$ *Rat-Exp*(Σ, K, Φ) *we define its semantics* $\|\zeta\| \in K\langle\langle T_\Sigma \rangle\rangle$ *simultaneously.*

- *For every* $a \in \Sigma_0$, *the expression* $a \in R$ *and* $\|a\| = 1a$,
- *for every* $k \geq 1, \sigma \in \Sigma_k$, *and* $\zeta_1, \ldots, \zeta_k \in R$, *the expression* $\sigma_\Phi(\zeta_1, \ldots, \zeta_k) \in R$ *and* $\|\sigma_\Phi(\zeta_1, \ldots, \zeta_k)\| = \sigma_\Phi(\|\zeta_1\|, \ldots, \|\zeta_k\|)$,
- *for every* $\zeta \in R$ *and* $k \in K$, *the expression* $k\zeta \in R$ *and* $\|k\zeta\| = k \cdot \|\zeta\|$,
- *for every* $\zeta_1, \zeta_2 \in R$, *the expression* $\zeta_1 + \zeta_2 \in R$ *and* $\|\zeta_1 + \zeta_2\| = \|\zeta_1\| + \|\zeta_2\|$,
- *for every* $\zeta_1, \zeta_2 \in R$ *and* $a \in \Sigma_0$, *the expression* $\zeta_1 \cdot_{a,\Phi} \zeta_2 \in R$ *and* $\|\zeta_1 \cdot_{a,\Phi} \zeta_2\| = \|\zeta_1\| \cdot_{a,\Phi} \|\zeta_2\|$, *and*
- *for every* $\zeta \in R$ *and* $a \in \Sigma_0$ *such that* $\|\zeta\|$ *is* a*-proper, the expression* $\zeta^*_{a,\Phi} \in R$ *and* $\left\|\zeta^*_{a,\Phi}\right\| = \|\zeta\|^*_{a,\Phi}$.

A tree series $S \in K\langle\langle T_\Sigma \rangle\rangle$ is called Φ-*rational over* Σ *and* K if there is a $\zeta \in$ *Rat-Exp*(Σ, K, Φ) such that $S = \|\zeta\|$. The class of all Φ-rational tree series over Σ and K is denoted by *Rat*(Σ, K, Φ). Clearly, the first four conditions in the above definition imply that $K\langle T_\Sigma \rangle \subseteq$ *Rat*(Σ, K, Φ). Moreover, *Rat*(Σ, K, Φ) is the smallest subclass of $K\langle\langle T_\Sigma \rangle\rangle$ which has this property and is closed under the Φ-rational operations on tree series.

Next, we wish to establish a Kleene theorem showing the coincidence of Φ-recognizable and Φ-rational tree series. For this, we shall need the subsequent lemma.

Lemma 5. *(cf. [15], Lm. 5.1) Consider a Φ-wta $\mathcal{M} = (Q, wt, ter)$. Let $P \subseteq Q, q \in Q$, and $p \in Q \setminus P$. Then*

$$\|\mathcal{M}\|(P \cup \{p\}, q) = \|\mathcal{M}\|(P, q) \cdot_{p,\Phi} \|\mathcal{M}\|(P, p)^*_{p,\Phi}.$$

Let Q be a finite set of nullary symbols with $Q \cap \Sigma = \emptyset$. Then $Rat(\Sigma \cup Q, K, \Phi)$ denotes the class of Φ-rational tree series over $\Sigma \cup Q$ and K defined by Φ-rational expressions from $Rat\text{-}Exp(\Sigma \cup Q, K, \Phi)$. We set

$$Rat(\Sigma + fin, K, \Phi) = \bigcup_{Q \text{ finite}} Rat(\Sigma \cup Q, K, \Phi)$$

and

$$Rec(\Sigma + fin, K, \Phi) = \bigcup_{Q \text{ finite}} Rec(\Sigma \cup Q, K, \Phi).$$

Now, we are ready to prove one half of our Kleene theorem.

Proposition 4. *(cf. [15], Thm. 5.2) $Rec(\Sigma, K, \Phi) \subseteq Rat(\Sigma + fin, K, \Phi)|_{T_\Sigma}$.*

Proof. Let $\mathcal{M} = (Q, wt, q_f)$ be a final weight normalized Φ-wta with $Q = \{q_1, \ldots, q_n\}$. We show that $\|\mathcal{M}\| \in Rat(\Sigma \cup Q, K, \Phi)$. Note that $(\|\mathcal{M}\|, t) = (\|\mathcal{M}\|(Q, q_f), t)$ for every $t \in T_\Sigma$. So

$$\|\mathcal{M}\| = (\ldots((\|\mathcal{M}\|(Q, q_f) \cdot_{\Phi,q_1} \mathbf{0}) \cdot_{\Phi,q_2} \mathbf{0}) \ldots) \cdot_{\Phi,q_n} \mathbf{0}|_{T_\Sigma}.$$

Thus it remains to prove that for every $P \subseteq Q$ and $q \in Q$, the tree series $\|\mathcal{M}\|(P, q) \in Rat(\Sigma \cup Q, K, \Phi)$. To this end, we apply induction on the number of elements of P. Let $P = \emptyset$. For every $k \geq 0, \sigma \in \Sigma_k$, and $p_1, \ldots, p_k \in Q$, we define the run $r^\sigma_{p_1,\ldots,p_k,q} : dom(\sigma(p_1, \ldots, p_k)) \to Q$ of $\mathcal{M}$ over $\sigma(p_1, \ldots, p_k)$ using $\emptyset$, such that $r^\sigma_{p_1,\ldots,p_k,q}(\varepsilon) = q$, and $r^\sigma_{p_1,\ldots,p_k,q}(i) = p_i$ for every $1 \leq i \leq k$. Then we have

$$R^\emptyset_\mathcal{M}(t, q) = \begin{cases} \{r^\sigma_{p_1,\ldots,p_k,q}\} & \text{if } t = \sigma(p_1, \ldots, p_k),\ k \geq 0, \sigma \in \Sigma_k, p_1, \ldots, p_k \in Q \\ \{r_q\} & \text{if } t = q \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that $(\|\mathcal{M}\|(\emptyset, q), q) = 0$ by definition. Thus $supp(\|\mathcal{M}\|(\emptyset, q)) \subseteq \Sigma(Q)$ where $\Sigma(Q) = \{\sigma(p_1, \ldots, p_k) \mid k \geq 0, \sigma \in \Sigma_k, p_1, \ldots, p_k \in Q\}$, i.e., $\|\mathcal{M}\|(\emptyset, q)$ is a polynomial, and hence a Φ-rational tree series.

For the induction step, assume that for every $q \in Q$ the tree series $\|\mathcal{M}\|(P, q)$ is Φ-rational over $\Sigma \cup Q$ and K. Let $p \in Q \setminus P$. Then, by Lemma 5 we get that $\|\mathcal{M}\|(P \cup \{p\}, q)$ is Φ-rational over $\Sigma \cup Q$ and K which in turn implies that $\|\mathcal{M}\| \in Rat(\Sigma \cup Q, K, \Phi)$. □

Example 4 (Example 3 continued). We shall construct a Φ-rational expression for the Φ-recognizable tree series $\|\mathcal{M}\|$ of Example 3 on page 417. Consider the next expressions

$$\zeta_0 = 0 + a, \quad \zeta_{q_i} = 0 + q_i \text{ for every } i = 1,2,3,$$
$$\zeta_1 = \left(\max_{p_1,p_2 \in Q} (0 + \gamma_\Phi(p_1,p_2)) \right)^*_{q_1,\Phi}, \quad \zeta_2 = \left(\max_{p_1,p_2 \in Q} (0 + \sigma_\Phi(p_1,p_2)) \right)^*_{q_2,\Phi},$$
$$\zeta_3 = 1 + \delta_\Phi(q_2,q_2,q_3), \quad \zeta_4 = \max_{\substack{p_1,p_2,p_3 \in Q \\ (p_1,p_2,p_3) \neq (q_2,q_2,q_3)}} (0 + \delta_\Phi(p_1,p_2,p_3)), \text{ and}$$
$$\zeta_5 = \max\left((\max(\zeta_3,\zeta_4))^*_{q_3,\Phi}, \zeta_{q_2} \right)$$

We have that $\|\mathcal{M}\|$ equals the restriction on T_Σ of the semantics of the Φ-rational expression

$$\left(\left(\left((\zeta_5 \cdot_{q_2,\Phi} \max(\zeta_2,\zeta_{q_1}) \cdot_{q_1,\Phi} \max(\zeta_1,\zeta_{q_3}))^*_{q_3,\Phi} \right) \cdot_{q_1,\Phi} \zeta_0 \right) \cdot_{q_2,\Phi} -\infty \right) \cdot_{q_3,\Phi} -\infty,$$

where we identify $-\infty$ and the constant tree series that takes all trees to $-\infty$.

In the sequel, we establish the inclusion $Rat(\Sigma,K,\Phi) \subseteq Rec(\Sigma,K,\Phi)$. For this, it suffices to show that the class $Rec(\Sigma,K,\Phi)$ contains the tree series $1a$ (for every $a \in \Sigma_0$) and is closed under the Φ-rational operations on tree series.

Lemma 6.

(i) (cf. [15], Lm. 6.1) For every $a \in \Sigma_0$, the tree series $1a \in Rec(\Sigma,K,\Phi)$.

(ii) (cf. [15], Lm. 6.2) The class $Rec(\Sigma,K,\Phi)$ is closed under Φ-top-concatenation.

(iii) (cf. [15], Lm. 6.5) Let $S_1, S_2 \in Rec(\Sigma,K,\Phi)$ and $a \in \Sigma_0$. Then the (a,Φ)-concatenation of S_2 and S_1 is a Φ-recognizable tree series, i.e., $S_2 \cdot_{a,\Phi} S_1 \in Rec(\Sigma,K,\Phi)$.

(iv) (cf. [15], Lm. 6.7) Let $a \in \Sigma_0$ and $S \in K^a\langle\langle T_\Sigma \rangle\rangle$ be Φ-recognizable. Then $S^*_{a,\Phi} \in Rec(\Sigma,K,\Phi)$.

Proof. (iii) Let $\mathcal{M}_1 = (Q_1, wt_1, q_{f_1})$ and $\mathcal{M}_2 = (Q_2, wt_2, q_{f_2})$ be final weight normalized Φ-wta with $\|\mathcal{M}_1\| = S_1$ and $\|\mathcal{M}_2\| = S_2$, and let us assume that $Q_1 \cap Q_2 = \emptyset$. We consider the final weight normalized Φ-wta $\mathcal{M} = (Q, wt, q_{f_2})$ with $Q = (Q_1 \cup Q_2) \setminus \{q_{f_1}\}$. For every $k \geq 0, \sigma \in \Sigma_k, q_1, \ldots, q_k, q \in Q$ we set

$$wt((q_1,\ldots,q_k),\sigma,q) = \begin{cases} wt_1((q_1,\ldots,q_k),\sigma,q) & \text{if } q_1,\ldots,q_k,q \in Q_1 \\ wt_1((q_1,\ldots,q_k),\sigma,q_{f_1}) \cdot wt_2(a,q) & \text{if } k \neq 0,\ q_1,\ldots,q_k \in Q_1, \text{and } q \in Q_2 \\ wt_2((q_1,\ldots,q_k),\sigma,q) & \text{if } k \neq 0,\ q_1,\ldots,q_k,q \in Q_2 \\ wt_2(\sigma,q) + wt_1(\sigma,q_{f_1}) \cdot wt_2(a,q) & \text{if } k = 0, \sigma \neq a, \text{ and } q \in Q_2 \\ wt_1(a,q_{f_1}) \cdot wt_2(a,q) & \text{if } k = 0, \sigma = a, \text{ and } q \in Q_2 \\ 0 & \text{otherwise.} \end{cases}$$

Then we can show that $\|\mathcal{M}\| = \|\mathcal{M}_2\| \cdot_{a,\Phi} \|\mathcal{M}_1\|$. □

Theorem 1. *(cf. [15], Thm. 6.8)*

(i) $Rec(\Sigma, K, \Phi)$ *is closed under the* Φ*-rational operations.*

(ii) $Rat(\Sigma, K, \Phi) \subseteq Rec(\Sigma, K, \Phi)$.

Let Q_∞ be an infinite set such that $Q \subseteq Q_\infty$ for every finite set Q. We define the operation $lift_\infty : \bigcup_{Q \text{ finite set}} K\langle\langle T_\Sigma(Q)\rangle\rangle \to K\langle\langle T_\Sigma(Q_\infty)\rangle\rangle$ in the following way. Let Q be a finite set and $S \in K\langle\langle T_\Sigma(Q)\rangle\rangle$. For every $t \in T_\Sigma(Q_\infty)$ we let

$$(lift_\infty(S), t) = \begin{cases} (S,t) & \text{if } t \in T_\Sigma(Q) \\ 0 & \text{otherwise.} \end{cases}$$

Now, we are ready to state our first main result, namely the Kleene theorem for Φ-recognizable tree series.

Theorem 2 (Kleene theorem). *(cf. [15], Thm. 7.1)*

$$lift_\infty(Rec(\Sigma + fin, K, \Phi)) = lift_\infty(Rat(\Sigma + fin, K, \Phi)).$$

Proof. By Theorem 1(ii) we have $Rat(\Sigma, K, \Phi) \subseteq Rec(\Sigma, K, \Phi)$. This implies that $Rat(\Sigma \cup Q, K, \Phi) \subseteq Rec(\Sigma \cup Q, K, \Phi)$, for every finite set Q. Therefore $Rat(\Sigma + fin, K, \Phi) \subseteq Rec(\Sigma + fin, K, \Phi)$ and thus

$$lift_\infty(Rat(\Sigma + fin, K, \Phi)) \subseteq lift_\infty(Rec(\Sigma + fin, K, \Phi)).$$

Conversely, let $S \in lift_\infty(Rec(\Sigma + fin, K, \Phi))$. Then, there is a finite set Q and $S' \in Rec(\Sigma \cup Q, K, \Phi)$ such that $S = lift_\infty(S')$. Then, by the proof of Proposition 4, there is another finite set Q' and a rational expression $\zeta \in Rat\text{-}Exp(\Sigma \cup Q \cup Q', K, \Phi)$ such that $\|\zeta\|_{|T_{\Sigma \cup Q}} = S'$ and for every $q \in Q'$, we have $(\|\zeta\|, q) = 0$. Then $lift_\infty(S') = lift_\infty(\|\zeta\|)$, hence $S = lift_\infty(\|\zeta\|) \in lift_\infty(Rat(\Sigma + fin, K, \Phi))$. □

5 Weighted MSO-logic with Φ-discounting over finite trees

In this section, we introduce a weighted monadic second-order logic (abbreviated to weighted MSO-logic) with Φ-discounting over finite trees, and characterize the class $Rec(\Sigma, K, \Phi)$ in terms of this logic. The syntax of our MSO-formulas is the one used in [18] but here we exclude second-order universal quantifiers since we do not need them for the description of our automata. For the semantics of our MSO-formulas, we employ the Φ-discounting. Let us first recall some basic terminology and definitions from [18].

Let $\mathcal{V}$ be a finite set of first and second-order variables. A tree $t \in T_\Sigma$ is represented by the structure $(dom(t), edge_1, \ldots, edge_{\deg(\Sigma)}, (label_\sigma)_{\sigma \in \Sigma})$ where for every $w, u \in dom(t)$ and $j \in \{1, \ldots, \deg(\Sigma)\}$, $edge_j(w,u)$ holds true iff $u = wj$ and $label_\sigma(w)$ holds true iff $t(w) = \sigma$. A $(t, \mathcal{V})$-*assignment* ρ is a mapping assigning

elements of $dom(t)$ to first order variables from $\mathcal{V}$, and subsets of $dom(t)$ to second-order variables from $\mathcal{V}$. If x is a first order variable and $w \in dom(t)$, then we denote by $\rho[x \to w]$ the $(t, \mathcal{V} \cup \{x\})$-assignment which associates w to x and acts as ρ on $\mathcal{V} \setminus \{x\}$. The notation $\rho[X \to I]$ for a second-order variable X and a set $I \subseteq dom(t)$ has a similar meaning.

In the rest of the paper, $\mathcal{V}$ will denote an arbitrary finite set of first and second-order variables.

Now, we consider the ranked alphabet $\Sigma_\mathcal{V} = \Sigma \times \{0,1\}^\mathcal{V}$ with $rk_{\Sigma_\mathcal{V}}(\sigma, f) = rk_\Sigma(\sigma)$ for every $\sigma \in \Sigma$ and $f \in \{0,1\}^\mathcal{V}$. For every $(\sigma, f) \in \Sigma_\mathcal{V}$ we denote by $(\sigma, f)_1$ and $(\sigma, f)_2$ the symbols σ and f, respectively. A tree $s \in T_{\Sigma_\mathcal{V}}$ is called *valid* if for every first order variable $x \in \mathcal{V}$, there is exactly one node w of s such that $(s(w)_2)(x) = 1$. The set of all valid finite trees over $\Sigma_\mathcal{V}$ is denoted by $T^v_{\Sigma_\mathcal{V}}$. Every valid tree $s \in T_{\Sigma_\mathcal{V}}$ corresponds to a pair (t, ρ) where $t \in T_\Sigma$ and ρ is a $(t, \mathcal{V})$-assignment, in the following way. It holds $dom(t) = dom(s)$ and $t(w) = s(w)_1$ for every $w \in dom(s)$, and for every first order variable x, second-order variable X, and every node $w \in dom(s)$, we have $\rho(x) = w$ iff $(s(w)_2)(x) = 1$, and $w \in \rho(X)$ iff $(s(w)_2)(X) = 1$. Then, we say that s and (t, ρ) *correspond to each other*. In the following, we identify every valid tree s with its corresponding pair (t, ρ).

Corollary 1. *The characteristic series $1_{T^v_{\Sigma_\mathcal{V}}} : T_{\Sigma_\mathcal{V}} \to K$ is Φ-recognizable.*

Let φ be an MSO-formula over trees [36, 37] with $Free(\varphi) \subseteq \mathcal{V}$. As usual we shall write Σ_φ for $\Sigma_{Free(\varphi)}$. For every $(t, \rho) \in T_{\Sigma_\mathcal{V}}$ we let $(t, \rho) \models \varphi$ whenever (t, ρ) satisfies φ (cf. [26]). The well-known result of Thatcher and Wright [35], and Doner [10] states that the tree language $\mathcal{L}_\mathcal{V}(\varphi) = \{(t, \rho) \in T^v_{\Sigma_\mathcal{V}} \mid (t, \rho) \models \varphi\}$ is recognizable; conversely, for every recognizable tree language $L \subseteq T_\Sigma$ there exists an MSO-sentence φ, such that $L = \mathcal{L}(\varphi)$ where $\mathcal{L}(\varphi) = \mathcal{L}_{Free(\varphi)}(\varphi)$.

Next we introduce our weighted MSO-logic with Φ-discounting over trees. For this we extend our Φ-discounting over Σ and K to a discounting over $\Sigma_\mathcal{V}$ and K. For simplicity we shall use the same symbol Φ. More precisely, for every $(\sigma, f) \in \Sigma_\mathcal{V}$ we set $\Phi_{(\sigma,f)} = \Phi_\sigma$.

Definition 4. *The set $MSO(\Sigma, K)$ of all formulas of the* weighted MSO-logic with Φ-discounting over Σ and K *on finite trees is defined to be the smallest set F such that*

- *F contains all atomic formulas $k, label_\sigma(x), edge_i(x, y), x \in X$ and the negations $\neg label_\sigma(x), \neg edge_i(x, y), \neg(x \in X)$, and*
- *if $\varphi, \psi \in F$, then also $\varphi \vee \psi, \varphi \wedge \psi, \exists x \,.\, \varphi, \exists X \,.\, \varphi, \forall x \,.\, \varphi \in F$,*

where $k \in K$, $\sigma \in \Sigma$, $1 \leq i \leq \deg(\Sigma)$, x, y are first order variables, and X is a second-order variable.

Next we define the semantics of the formulas in $MSO(\Sigma, K)$ as tree series in $K\langle\langle T_{\Sigma_\mathcal{V}} \rangle\rangle$. As in the word case [16], we employ the Φ-discounting only in the semantics of first order universal quantifications.

Definition 5. *Let $\varphi \in MSO(\Sigma, K)$ with $Free(\varphi) \subseteq \mathcal{V}$. The* Φ-semantics *of φ is a tree series $\|\varphi\|_{\mathcal{V}} \in K\langle\langle T_{\Sigma_{\mathcal{V}}}\rangle\rangle$ defined as follows. Let $s \in T_{\Sigma_{\mathcal{V}}}$. If s is not a valid tree, then $(\|\varphi\|_{\mathcal{V}}, s) = 0$. Otherwise, let ρ be a $(t, \mathcal{V})$-assignment such that s and (t, ρ) correspond to each other. Then, we inductively define $(\|\varphi\|_{\mathcal{V}}, s) \in K$ as follows:*

- $(\|k\|_{\mathcal{V}}, s) = k$
- $(\|label_\sigma(x)\|_{\mathcal{V}}, s) = \begin{cases} 1 & \text{if } t(\rho(x)) = \sigma \\ 0 & \text{otherwise} \end{cases}$
- $(\|edge_i(x,y)\|_{\mathcal{V}}, s) = \begin{cases} 1 & \text{if } \rho(y) = \rho(x)i \\ 0 & \text{otherwise} \end{cases}$
- $(\|x \in X\|_{\mathcal{V}}, s) = \begin{cases} 1 & \text{if } \rho(x) \in \rho(X) \\ 0 & \text{otherwise} \end{cases}$
- $(\|\neg\varphi\|_{\mathcal{V}}, s) = \begin{cases} 1 & \text{if } (\|\varphi\|_{\mathcal{V}}, s) = 0 \\ 0 & \text{if } (\|\varphi\|_{\mathcal{V}}, s) = 1 \end{cases}$, *provided that φ is of the form $label_\sigma(x)$, $edge_i(x,y)$, or $x \in X$*
- $(\|\varphi \vee \psi\|_{\mathcal{V}}, s) = (\|\varphi\|_{\mathcal{V}}, s) + (\|\psi\|_{\mathcal{V}}, s)$
- $(\|\varphi \wedge \psi\|_{\mathcal{V}}, s) = (\|\varphi\|_{\mathcal{V}}, s) \cdot (\|\psi\|_{\mathcal{V}}, s)$
- $(\|\exists x \,.\, \varphi\|_{\mathcal{V}}, s) = \sum_{w \in dom(t)} \left(\|\varphi\|_{\mathcal{V}\cup\{x\}}, s[x \to w]\right)$
- $(\|\exists X \,.\, \varphi\|_{\mathcal{V}}, s) = \sum_{I \subseteq dom(t)} \left(\|\varphi\|_{\mathcal{V}\cup\{X\}}, s[X \to I]\right)$
- $(\|\forall x \,.\, \varphi\|_{\mathcal{V}}, s) = \prod_{w \in dom(t)} \Phi_w^t\left(\left(\|\varphi\|_{\mathcal{V}\cup\{x\}}, s[x \to w]\right)\right)$.

We shall simply write $\|\varphi\|$ for $\|\varphi\|_{Free(\varphi)}$. If φ has no free variables, i.e., if it is a sentence, then $\|\varphi\| \in K\langle\langle T_\Sigma\rangle\rangle$. One should observe that the Φ-semantics $\|\varphi\|_{\mathcal{V}}$ of every formula $\varphi \in MSO(\Sigma, K)$ is defined according to a finite set of variables $\mathcal{V}$ containing $Free(\varphi)$. Actually, this is not an essential restriction as it is announced in the subsequent proposition.

Proposition 5. *(cf. [11], Prop. 3.3) Let $\varphi \in MSO(\Sigma, K)$ with $Free(\varphi) \subseteq \mathcal{V}$. Then*

$$(\|\varphi\|_{\mathcal{V}}, s) = \left(\|\varphi\|, s|_{Free(\varphi)}\right)$$

for every $s \in T^v_{\Sigma_{\mathcal{V}}}$. Moreover, the tree series $\|\varphi\|$ is Φ-recognizable (resp. a recognizable step function) over Σ_φ iff $\|\varphi\|_{\mathcal{V}}$ is Φ-recognizable (resp. a recognizable step function) over $\Sigma_{\mathcal{V}}$.

Definition 6. *(i) A formula $\varphi \in MSO(\Sigma, K)$ is called* restricted *if whenever φ contains a universal first order quantification $\forall x.\psi$, then $\|\psi\|$ is a recognizable step function.*

(ii) A formula $\varphi \in MSO(\Sigma, K)$ is called almost existential *if whenever φ contains a universal first order quantification $\forall x \,.\, \psi$ and ψ contains a universal first order quantification $\forall y.\psi'$, then ψ' is composed from conjunctions of negations of atomic formulas of the form $edge_i(z, z')$, where $1 \leq i \leq \deg(\Sigma)$.*

We denote by $RMSO(\Sigma, K)$ the class of all restricted formulas of $MSO(\Sigma, K)$, and by $REMSO(\Sigma, K)$ the class of all restricted existential $MSO(\Sigma, K)$-formulas, i.e., formulas of the form $\exists X_1 \ldots \exists X_n \,.\, \psi$ with $\psi \in RMSO(\Sigma, K)$ containing no set quantification. Furthermore, we let $AEMSO(\Sigma, K)$ for the class of all almost existential formulas of $MSO(\Sigma, K)$. A tree series $S \in K\langle\langle T_\Sigma \rangle\rangle$ is called *RMSO-Φ-definable* (resp. *REMSO-Φ-definable*, *AEMSO-Φ-definable*) if there is a sentence $\varphi \in RMSO(\Sigma, K)$ (resp. $\varphi \in REMSO(\Sigma, K)$, $\varphi \in AEMSO(\Sigma, K)$) such that $S = \|\varphi\|$. We let $r\text{-}Mso(\Sigma, K, \Phi)$ (resp. $er\text{-}Mso(\Sigma, K, \Phi)$, $ae\text{-}Mso(\Sigma, K, \Phi)$) comprise all RMSO-Φ-definable (resp. REMSO-Φ-definable, AEMSO-Φ-definable) tree series over Σ and K.

Our second main result is the following.

Theorem 3. *(i) $Rec(\Sigma, K, \Phi) = r\text{-}Mso(\Sigma, K, \Phi) = er\text{-}Mso(\Sigma, K, \Phi)$.*

(ii) If K is additively locally finite, then $Rec(\Sigma, K, \Phi) = ae\text{-}Mso(\Sigma, K, \Phi)$.

For the proof, we firstly show by induction on the structure of formulas φ that $r\text{-}Mso(\Sigma, K, \Phi) \subseteq Rec(\Sigma, K, \Phi)$, and whenever K is additively locally finite, then $ae\text{-}Mso(\Sigma, K, \Phi) \subseteq Rec(\Sigma, K, \Phi)$. This is incorporated in the subsequent lemma.

Lemma 7. *Let $\varphi, \psi \in MSO(\Sigma, K)$. Then*

(i) (cf. [18], Lm. 5.2) if φ is an atomic formula or the negation of an atomic formula, then $\|\varphi\|$ is a recognizable step function,

(ii) (cf. [18], Lm. 5.3, and [16], Lm. 13) if $\|\varphi\|, \|\psi\|$ are Φ-recognizable (resp. recognizable step functions), then $\|\varphi \vee \psi\|$ and $\|\varphi \wedge \psi\|$ are Φ-recognizable (resp. recognizable step functions),

(iii) (cf. [18], Lm. 5.4) if $\|\varphi\|$ is Φ-recognizable, then $\|\exists x \,.\, \varphi\|$ and $\|\exists X \,.\, \varphi\|$ are Φ-recognizable,

(iv) if K is additively locally finite and $\|\varphi\|$ is a recognizable step function, then $\|\exists x \,.\, \varphi\|$ and $\|\exists X \,.\, \varphi\|$ are recognizable step functions,

(v) if $\|\varphi\|$ is a recognizable step function, then $\|\forall x \,.\, \varphi\|$ is Φ-recognizable, and

(vi) if $\|\varphi\| = 1_L$, where $L \subseteq T^v_{\Sigma_\varphi}$ is a recognizable tree language, then $\|\forall x \,.\, \varphi\|$ is a recognizable step function.

Proof. (iv) We follow the proof of Lemma 17 in [16] using our Proposition 2(iii) on page 421.

(v) Let $\mathcal{W} = Free(\varphi) \cup \{x\}$ and $\mathcal{V} = Free(\forall x \,.\, \varphi) = \mathcal{W} \setminus \{x\}$. By Proposition 5 (in case $x \notin Free(\varphi)$) let $\|\varphi\|_{\mathcal{W}} = \sum_{j=1}^{n} k_j 1_{L_j}$, where $k_j \in K$ and $L_j \subseteq T^v_{\Sigma_{\mathcal{W}}}$ are

recognizable tree languages ($1 \leq j \leq n$). Furthermore, we assume that the family $(L_j)_{1\leq j\leq n}$ is a partition of $T^v_{\Sigma_\mathcal{W}}$.

Let $\widetilde{\Sigma} = \Sigma \times \{1,\ldots,n\}$ be the ranked alphabet with $rk_{\widetilde{\Sigma}}((\sigma,j)) = rk_\Sigma(\sigma)$ for every $(\sigma,j) \in \widetilde{\Sigma}$. Every tree $s \in T^v_{\widetilde{\Sigma}_\mathcal{V}}$ can be written as a triple (t,v,ρ) where $(t,\rho) \in T^v_{\Sigma_\mathcal{V}}$, $dom(t) = dom(s)$, and v is a mapping $v : dom(s) \to \{1,\ldots,n\}$ determined by $v(w) = j$ whenever $s(w) = (\sigma,j,f)$ for some $\sigma \in \Sigma$ and $f \in \{0,1\}^\mathcal{V}$. Conversely, every such triple (t,v,ρ) corresponds to a tree $s \in T^v_{\widetilde{\Sigma}_\mathcal{V}}$. Hence in the sequel, we write the elements of $T^v_{\widetilde{\Sigma}_\mathcal{V}}$ in the form (t,v,ρ). Let $\widetilde{L}$ be the set of all trees $(t,v,\rho) \in T^v_{\widetilde{\Sigma}_\mathcal{V}}$ such that for every $w \in dom(t)$ and $1 \leq j \leq n$ if $v(w) = j$, then $(t,\rho[x \to w]) \in L_j$.

Since $(L_j)_{1\leq j\leq n}$ is a partition of $T^v_{\Sigma_\mathcal{W}}$, for every $(t,\rho) \in T^v_{\Sigma_\mathcal{V}}$ there is a unique $v : dom(t) \to \{1,\ldots,n\}$ such that $(t,v,\rho) \in \widetilde{L}$. By Lemma 5.5 in [18], we get that $\widetilde{L}$ is recognizable. Let $\widetilde{\mathcal{M}} = \left(Q, \widetilde{\Sigma}_\mathcal{V}, \delta, F\right)$ be a deterministic bottom-up tree automaton accepting $\widetilde{L}$. We consider the Φ-wta $\mathcal{M} = (Q, wt, ter)$ over $\widetilde{\Sigma}_\mathcal{V}$ and K with weight assignment mapping wt defined for every $m \geq 0, (\sigma,j,f) \in \left(\widetilde{\Sigma}_\mathcal{V}\right)_m$, and $q_1,\ldots,q_m,q \in Q$ by

$$wt((q_1,\ldots,q_m),(\sigma,j,f),q) = \begin{cases} k_j & \text{if } \delta_{(\sigma,j,f)}(q_1,\ldots,q_m) = q \\ 0 & \text{otherwise.} \end{cases}$$

The final distribution ter is determined by $ter(q) = 1$ if $q \in F$, and $ter(q) = 0$ otherwise for every $q \in Q$.

Since $\widetilde{\mathcal{M}}$ is deterministic, for every $(t,v,\rho) \in T^v_{\widetilde{\Sigma}_\mathcal{V}}$ there is at most one run $r_{(t,v,\rho)}$ of $\widetilde{\mathcal{M}}$ over (t,v,ρ). Moreover, since $\left|\widetilde{\mathcal{M}}\right| = \widetilde{L}$ we get

$$\left(\|\mathcal{M}\|,(t,v,\rho)\right) = \begin{cases} \prod\limits_{w\in dom((t,v,\rho))} \Phi_w^{(t,v,\rho)}\left(wt\left(r_{(t,v,\rho)},w\right)\right) & \text{if } (t,v,\rho) \in \widetilde{L} \\ 0 & \text{otherwise.} \end{cases}$$

Let $(t,v,\rho) \in \widetilde{L}$. For every $w \in dom(t)$ with $v(w) = j$, we have $wt\left(r_{(t,v,\rho)},w\right) = k_j$, and $(t,\rho[x \to w]) \in L_j$ which in turn implies that $\left(\|\varphi\|_{\mathcal{V}\cup\{x\}},(t,\rho[x \to w])\right) = k_j$. We consider the relabeling $h : \widetilde{\Sigma}_\mathcal{V} \to \Sigma_\mathcal{V}$ by $h((\sigma,j,f)) = (\sigma,f)$ for every $(\sigma,j,f) \in \widetilde{\Sigma}_\mathcal{V}$. Then for every $(t,\rho) \in T^v_{\Sigma_\mathcal{V}}$,

$$\begin{aligned}
\left(h\left(\|\mathcal{M}\|\right),(t,\rho)\right) &= \sum_{(t,v,\rho)\in h^{-1}((t,\rho))} \left(\|\mathcal{M}\|,(t,v,\rho)\right) = \left(\|\mathcal{M}\|,(t,v,\rho)\right) \\
&\quad \text{(where } (t,v,\rho) \in \widetilde{L}\text{)} \\
&= \prod_{w\in dom((t,v,\rho))} \Phi_w^{(t,v,\rho)}\left(wt\left(r_{(t,v,\rho)},w\right)\right). \\
&= \prod_{w\in dom(t)} \Phi_w^t\left(\left(\|\varphi\|_{\mathcal{V}\cup\{x\}},(t,\rho[x \to w])\right)\right) = \left(\|\forall x \cdot \varphi\|,(t,\rho)\right).
\end{aligned}$$

Therefore, $\|\forall x \,.\, \varphi\| = h(\|\mathcal{M}\|)$ which by Proposition 1 on page 420 is Φ-recognizable. □

Proposition 6.

- $r\text{-}Mso(\Sigma, K, \Phi) \subseteq Rec(\Sigma, K, \Phi)$.
- *If K is additively locally finite, then* $ae\text{-}Mso(\Sigma, K, \Phi) \subseteq Rec(\Sigma, K, \Phi)$.

Proposition 7. $Rec(\Sigma, K, \Phi) \subseteq er\text{-}Mso(\Sigma, K, \Phi) \cap ae\text{-}Mso(\Sigma, K, \Phi)$.

Proof of Theorem 3. It is immediate by Propositions 6 and 7. □

6 Weighted Muller tree automata with Φ-discounting

In this section, we investigate weighted Muller tree automata with Φ-discounting acting on infinite trees. The underlying semiring is the max-plus semiring $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \sup, +, -\infty, 0)$, where we consider sup instead of max since we need to compute over infinite trees. Our results can be applied to the min-plus semiring $(\mathbb{R}_+ \cup \{\infty\}, \inf, +, \infty, 0)$ as well. A weighted Muller tree automaton with Φ-discounting computes the weight of a run (of an input infinite tree) by applying the Φ-discounting over $\mathbb{R}_{\max}$. By considering suitable endomorphisms for Φ, we do not require any completeness axioms (for the sum operation) in $\mathbb{R}_{\max}$ (cf. [32]). For a study on weighted Muller automata with Φ-discounting over infinite words cf. [16].

An *infinitary tree series S over Σ and* $\mathbb{R}_{\max}$ is a mapping $S : T^\omega_\Sigma \to \mathbb{R}_{\max}$. The class of all infinitary tree series over Σ and $\mathbb{R}_{\max}$ is denoted by $\mathbb{R}_{\max}\langle\langle T^\omega_\Sigma \rangle\rangle$. Let $S \in \mathbb{R}_{\max}\langle\langle T^\omega_\Sigma \rangle\rangle$. The *image* $Im(S)$ *of* S is the set $Im(S) = \{k \in \mathbb{R}_+ \cup \{-\infty\} \mid \exists t \in T^\omega_\Sigma$ with $(S,t) = k\}$. We say that S has *bounded image* if there is an $m \in \mathbb{R}_+$ such that $k \leq m$ for every $k \in Im(S)$. Consider an infinitary tree language $L \subseteq T^\omega_\Sigma$. The *characteristic series* $0_L : T^\omega_\Sigma \to \mathbb{R}_{\max}$ *of* L is defined in a similar way as for finitary tree languages. Furthermore, for $S, T \in \mathbb{R}_{\max}\langle\langle T^\omega_\Sigma \rangle\rangle$ and $k \in \mathbb{R}_{\max}$, the sum, the scalar product, and the Hadamard product are now written as $\max(S,T)$, $k + S$, and $S + T$, respectively, and defined in the obvious way.

Let $h : \Sigma \to \Gamma$ be a relabeling. Then h is extended to a mapping $h : T^\omega_\Sigma \to T^\omega_\Gamma$ such that $dom(h(t)) = dom(t)$ and $h(t)(w) = h(t(w))$ for every $t \in T^\omega_\Sigma$ and $w \in dom(t)$. Moreover, h can be extended to a partial mapping $h : \mathbb{R}_{\max}\langle\langle T^\omega_\Sigma \rangle\rangle \to \mathbb{R}_{\max}\langle\langle T^\omega_\Gamma \rangle\rangle$ in the following way. For every $S \in \mathbb{R}_{\max}\langle\langle T^\omega_\Sigma \rangle\rangle$ with bounded image, we define the series $h(S) \in \mathbb{R}_{\max}\langle\langle T^\omega_\Gamma \rangle\rangle$ by $(h(S), s) = \sup\{(S,t) \mid t \in h^{-1}(s)\}$ for every $s \in T^\omega_\Gamma$. Furthermore, for every $T \in \mathbb{R}_{\max}\langle\langle T^\omega_\Gamma \rangle\rangle$, the series $h^{-1}(T) \in \mathbb{R}_{\max}\langle\langle T^\omega_\Sigma \rangle\rangle$ is determined by $(h^{-1}(T), t) = (T, h(t))$ for every $t \in T^\omega_\Sigma$.

Let $\Phi = (\Phi_k)_{k \geq 1}$ be a discounting over Σ and $\mathbb{R}_{\max}$. Recall (cf. [14], and Example 1 on page 414) that every endomorphism of $\mathbb{R}_{\max}$ is of the form $\overline{p}$:

$\mathbb{R}_{\max} \to \mathbb{R}_{\max}$ where $p \in \mathbb{R}_+$ and $x \longmapsto p \cdot x$ for every $x \in \mathbb{R}_+ \cup \{-\infty\}$ with the convention that $p \cdot (-\infty) = (-\infty) \cdot p = -\infty$ for every $p \in \mathbb{R}_+ \cup \{-\infty\}$. For our Φ-discounting here, we require for every $k \geq 1, \sigma \in \Sigma_k$ that $\Phi^i_\sigma = \overline{p^i_\sigma}$ with $0 \leq p^i_\sigma < 1$. Then, we simply write $\Phi_\sigma = \overline{p_\sigma} = \left(\overline{p^1_\sigma}, \ldots, \overline{p^k_\sigma}\right)$ for every $k \geq 1$ and $\sigma \in \Sigma_k$. Furthermore, for every $t \in T^\omega_\Sigma$ and every $w \in dom(t)$ we write $\overline{p^t_w}$ for Φ^t_w where

$$p^t_w = \begin{cases} 1 & \text{if } w = \varepsilon \\ p^{i_1}_{t(\varepsilon)} \cdot p^{i_2}_{t(i_1)} \cdot \ldots \cdot p^{i_n}_{t(i_1 \ldots i_{n-1})} & \text{if } w = i_1 \ldots i_n \text{ with } i_1, \ldots, i_n \in \mathbb{N}_+, n > 0. \end{cases}$$

We let $m_\Phi = \max\left\{p^i_\sigma \mid k \geq 1, \sigma \in \Sigma_k, \text{ and } 1 \leq i \leq k\right\}$. In the sequel, we shall use also the concatenation notation for the multiplication in $\mathbb{R}_+ \cup \{-\infty\}$.

Definition 7. *A* weighted Muller tree automaton with Φ-discounting *(*Φ-wmta *for short)* over Σ and $\mathbb{R}_{\max}$ *is a quadruple* $\mathcal{M} = (Q, in, wt, \mathcal{F})$, *where* Q *is the* finite state set, $in : Q \to \mathbb{R}_{\max}$ *is the* initial distribution, $wt : \bigcup_{k \geq 0} Q \times \Sigma_k \times Q^k \to \mathbb{R}_{\max}$ *is the* mapping assigning weights to the transitions *of the automaton, and* $\mathcal{F} \subseteq \mathcal{P}(Q)$ *is the* family of final states sets.

Let $t \in T^\omega_\Sigma$. A *run of* $\mathcal{M}$ *over* t is a mapping $r_t : dom(t) \to Q$. The *weight of* r_t *at* $w \in dom(t)$ is the value

$$wt(r_t, w) = wt(r_t(w), t(w), (r_t(w1), \ldots, r_t(w.rk_\Sigma(t(w))))).$$

The Φ-*weight* (or simply *weight*) *of* r_t, which is denoted by $weight_{\mathcal{M}}(r_t)$ (or simply $weight(r_t)$), is defined by

$$weight_{\mathcal{M}}(r_t) = in(r_t(\varepsilon)) + \sum_{w \in dom(t)} \frac{p^t_w}{\deg(\Sigma)^{|w|}} \cdot wt(r_t, w).$$

One should observe that in comparison to the finitary case, here we divide every summand of the infinite sum with a power of $\deg(\Sigma)$. This is needed to achieve the convergence of the infinite sum. Indeed, let $M = \max\left\{wt(\tau) \mid \tau \in \bigcup_{k \geq 0} Q \times \Sigma_k \times Q^k\right\}$. Then we have

$$\sum_{w \in dom(t)} \frac{p^t_w}{\deg(\Sigma)^{|w|}} \cdot wt(r_t, w) \leq M \cdot \sum_{n \geq 0} \sum_{\substack{w \in dom(t) \\ |w| = n}} \frac{m^{|w|}_\Phi}{\deg(\Sigma)^{|w|}}$$

$$\leq M \cdot \sum_{n \geq 0} \deg(\Sigma)^n \frac{m^{n \cdot}_\Phi}{\deg(\Sigma)^n} = M \cdot \frac{1}{1 - m_\Phi}.$$

Every infinite prefix-closed chain $\pi \subseteq dom(t)$ is called an *infinite path of* t. The run r_t is called *successful* if for every infinite path π of t, the set of states that appear infinitely often along π, constitutes a final state set. We shall denote by $R_{\mathcal{M}}(t)$ the set of all runs of $\mathcal{M}$ over t, and by $R^{suc}_{\mathcal{M}}(t)$ the set of all successful runs in $R_{\mathcal{M}}(t)$.

The Φ-*behavior* (or simply *behavior*) *of* $\mathcal{M}$ is the infinitary tree series $\|\mathcal{M}\| \in \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Sigma}\rangle\rangle$ whose coefficients are determined for every $t \in T^{\omega}_{\Sigma}$ by

$$(\|\mathcal{M}\|, t) = \sup\{(weight_{\mathcal{M}}(r_t)) \mid r_t \in R^{suc}_{\mathcal{M}}(t)\}.$$

Clearly, this supremum exists in $\mathbb{R}_{\max}$ since the values $weight_{\mathcal{M}}(r_t)$ are bounded by $N + M \cdot \frac{1}{1-m_\Phi}$, where $N = \max\{in(q) \mid q \in Q\}$.

A tree series $S \in \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Sigma}\rangle\rangle$ is said to be (Φ,ω)-*recognizable* (or Φ-*Muller recognizable*) if there exists a Φ-wmta $\mathcal{M}$ over Σ and $\mathbb{R}_{\max}$ such that $S = \|\mathcal{M}\|$. The family of all (Φ,ω)-recognizable tree series over Σ and $\mathbb{R}_{\max}$ is denoted by ω-$Rec(\Sigma, \mathbb{R}_{\max}, \Phi)$. Clearly, every (Φ,ω)-recognizable tree series $S \in \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Sigma}\rangle\rangle$ has bounded image. The next proposition collects closure properties of (Φ,ω)-recognizable tree series.

Proposition 8. *(i) (cf. [32], Prop. 8) The class* ω-$Rec(\Sigma, \mathbb{R}_{\max}, \Phi)$ *is closed under sum, scalar product, and Hadamard product.*

(ii) (cf. [32], Prop. 9) Let $h : \Sigma \to \Gamma$ *be a relabeling. Furthermore, for the* Φ*-discounting over* Σ *and* $\mathbb{R}_{\max}$ *we assume that* $\overline{p_\sigma} = \overline{p_{\sigma'}}$ *whenever* $h(\sigma) = h(\sigma')$ *for every* $\sigma, \sigma' \in \Sigma_k$*, and* $k \geq 1$*. Let* $\Phi' = (\Phi'_k)_{k\geq 1}$ *be the discounting over* Γ *and* $\mathbb{R}_{\max}$ *determined for every* $\gamma \in \Gamma_k$ *(*$k \geq 1$*) by* $\overline{p'_\gamma} = \overline{p_\sigma}$ *for every* $\sigma \in \Sigma_k$ *(*$k \geq 1$*) with* $h(\sigma) = \gamma$*. If* $S \in \omega$-$Rec(\Sigma, \mathbb{R}_{\max}, \Phi)$*, then* $h(S) \in \omega$-$Rec(\Gamma, \mathbb{R}_{\max}, \Phi')$*. Furthermore, if* $T \in \omega$-$Rec(\Gamma, \mathbb{R}_{\max}, \Phi')$*, then* $h^{-1}(T) \in \omega$-$Rec(\Sigma, \mathbb{R}_{\max}, \Phi)$*.*

(iii) (cf. [32], Prop. 10) Let $L \subseteq T^{\omega}_{\Sigma}$ *be an* ω*-recognizable tree language. Then* $0_L \in \omega$-$Rec(\Sigma, \mathbb{R}_{\max}, \Phi)$*.*

An infinitary tree series $S \in \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Sigma}\rangle\rangle$ is called an ω-*recognizable step function* (or *Muller recognizable step function*) if $S = \max_{1\leq j\leq n}(k_j + 0_{L_j})$ where $k_j \in \mathbb{R}_+ \cup \{-\infty\}$ and L_j is an ω-recognizable tree language for every $1 \leq j \leq n$.

Proposition 9. *(i) (cf. [32], Prop. 11) The class of* ω*-recognizable step functions over* Σ *and* $\mathbb{R}_{\max}$ *is closed under sum, scalar product, and Hadamard product.*

(ii) (cf. [32], Prop. 12) Let $h : \Sigma \to \Gamma$ *be a relabeling. Then* $h : \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Sigma}\rangle\rangle \to \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Gamma}\rangle\rangle$ *and* $h^{-1} : \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Gamma}\rangle\rangle \to \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Sigma}\rangle\rangle$ *preserve* ω*-recognizable step functions.*

7 Weighted MSO-logic with Φ-discounting over infinite trees

In this section we deal with weighted MSO-logic with Φ-discounting over the semiring $\mathbb{R}_{\max}$, and we interpret the semantics of weighted MSO-formulas as formal series over infinite trees. In our logic here, we inlude the atomic formula $x = y$ and its negation as well as second-order universal quantifiers (cf. [32]).

Every infinite tree $t \in T_\Sigma^\omega$ is represented by the structure $(dom(t), edge_1, \ldots, edge_{\deg(\Sigma)}, (label_\sigma)_{\sigma\in\Sigma})$. The notions of a $(t,\mathcal{V})$-assignment and the set $T_{\Sigma_\mathcal{V}}^{\omega,v}$ of all valid infinite trees over $\Sigma_\mathcal{V}$ are defined as in the case of finite trees. Similarly, every valid tree $s \in T_{\Sigma_\mathcal{V}}^{\omega,v}$ corresponds to a pair (t,ρ) where $t \in T_\Sigma^\omega$ and ρ is a $(t,\mathcal{V})$-assignment. The infinitary tree language $T_{\Sigma_\mathcal{V}}^{\omega,v}$ is ω-recognizable (cf. [32]), and thus the characteristic series $0_{T_{\Sigma_\mathcal{V}}^{\omega,v}} \in \mathbb{R}_{\max}\langle\langle T_{\Sigma_\mathcal{V}}^\omega \rangle\rangle$ is (Φ,ω)-recognizable.

Let φ be an MSO-formula [36, 37] over trees. Then for $Free(\varphi) \subseteq \mathcal{V}$, the well-known result of Rabin [29] states that the tree language $\mathcal{L}_\mathcal{V}^\omega(\varphi) = \{(t,\rho) \in T_{\Sigma_\mathcal{V}}^{\omega,v} \mid (t,\rho) \models \varphi\}$ is ω-recognizable; conversely, for every ω-recognizable tree language $L \subseteq T_\Sigma^\omega$ there exists an MSO-sentence φ, such that $L = \mathcal{L}^\omega(\varphi)$, where we simply write $\mathcal{L}^\omega(\varphi)$ for $\mathcal{L}^\omega_{Free(\varphi)}(\varphi)$.

Definition 8. *The set $MSO(\Sigma,\mathbb{R}_{\max})$ of all formulas of the weighted MSO-logic with Φ-discounting over Σ and $\mathbb{R}_{\max}$ on infinite trees is defined to be the smallest set F such that*

- *F contains all atomic formulas $k, label_\sigma(x), edge_i(x,y), x = y, x \in X$ and the negations $\neg label_\sigma(x), \neg edge_i(x,y), \neg(x = y), \neg(x \in X)$,*
- *if $\varphi, \psi \in F$, then $\varphi \vee \psi, \varphi \wedge \psi, \exists x \cdot \varphi, \exists X \cdot \varphi, \forall x \cdot \varphi \in F$, and if φ does not contain any constant $k \in \mathbb{R}_+ \setminus \{0\}$, then $\forall X \cdot \varphi \in F$*

where $k \in \mathbb{R}_+ \cup \{-\infty\}$, $\sigma \in \Sigma$, $1 \le i \le \deg(\Sigma)$, x, y are first order variables, and X is a second-order variable.

Next we define the semantics of the formulas in $MSO(\mathbb{R}_{\max},\Sigma)$ as infinitary tree series in $\mathbb{R}_{\max}\langle\langle T_{\Sigma_\mathcal{V}}^\omega \rangle\rangle$.

Definition 9. *Let $\varphi \in MSO(\Sigma,\mathbb{R}_{\max})$ and $Free(\varphi) \subseteq \mathcal{V}$. The Φ-semantics of φ is an infinitary tree series $\|\varphi\|_\mathcal{V} \in \mathbb{R}_{\max}\langle\langle T_{\Sigma_\mathcal{V}}^\omega \rangle\rangle$ defined as follows. Let $s \in T_{\Sigma_\mathcal{V}}^\omega$. If s is not a valid tree, then $(\|\varphi\|_\mathcal{V}, s) = -\infty$. Otherwise, let ρ be a $(t,\mathcal{V})$-assignment such that s and (t,ρ) correspond to each other. Then, we inductively define $(\|\varphi\|_\mathcal{V}, s)$ as in Definition 5, where $K = \mathbb{R}_{\max}$, except for the formulas $x = y$, $\forall x \cdot \varphi$, and $\forall X \cdot \varphi$ where we set*

- $(\|x = y\|_\mathcal{V}, s) = \begin{cases} 0 & \text{if } \rho(x) = \rho(y) \\ -\infty & \text{otherwise} \end{cases}$

- $(\|\forall x \cdot \varphi\|_\mathcal{V}, s) = \sum_{w \in dom(t)} \frac{p_w^t}{\deg(\Sigma)^{|w|}} \left(\|\varphi\|_{\mathcal{V}\cup\{x\}}, s[x \to w]\right)$.

$$- \left(\|\forall X \cdot \varphi\|_{\mathcal{V}}, s\right) = \sum_{I \subseteq dom(t)} \left(\|\varphi\|_{\mathcal{V}\cup\{X\}}, s[X \to I]\right).$$

Note that in the above definition of the semantics, the sums and products in Definition 5 on page 428 are replaced respectively, by suprema and sums in $\mathbb{R}_{\max}$. Moreover, in case of $\forall X \cdot \varphi$ the infinite sum is well-defined, because by definition the semantics of φ takes on only the values 0 and $-\infty$. We shall simply write $\|\varphi\|$ for $\|\varphi\|_{Free(\varphi)}$. As in the case of finitary tree series we can show the subsequent result.

Proposition 10. *Let $\varphi \in MSO(\Sigma, \mathbb{R}_{\max})$ with $Free(\varphi) \subseteq \mathcal{V}$. Then*

$$\left(\|\varphi\|_{\mathcal{V}}, s\right) = \left(\|\varphi\|, s|_{Free(\varphi)}\right)$$

for every $s \in T^{\omega,v}_{\Sigma_{\mathcal{V}}}$. Moreover, the tree series $\|\varphi\|$ is (Φ,ω)-recognizable (resp. an ω-recognizable step function) over Σ_φ iff $\|\varphi\|_{\mathcal{V}}$ is (Φ,ω)-recognizable (resp. an ω-recognizable step function) over $\Sigma_{\mathcal{V}}$.

Definition 10. *A formula $\varphi \in MSO(\Sigma, \mathbb{R}_{\max})$ is called* restricted *if whenever φ contains a universal first order quantification $\forall x \cdot \psi$, then $\|\psi\|$ is an ω-recognizable step function.*

Definition 11. *A formula $\varphi \in MSO(\Sigma, \mathbb{R}_{\max})$ is called* incomplete universal *if whenever φ contains a subformula $\forall x \cdot \psi$ such that ψ contains universal quantifiers, then ψ cannot contain any constant $k \in \mathbb{R}_+ \setminus \{0\}$.*

We denote by $RMSO(\Sigma, \mathbb{R}_{\max})$ (resp. $IUMSO(\Sigma, \mathbb{R}_{\max})$) the class of all restricted (resp. incomplete universal) formulas of $MSO(\Sigma, \mathbb{R}_{\max})$. A tree series $S \in \mathbb{R}_{\max}\langle\langle T^{\omega}_{\Sigma}\rangle\rangle$ is called *RMSO-Φ-definable* (resp. *IUMSO-Φ-definable*) if there is a sentence $\varphi \in RMSO(\Sigma, \mathbb{R}_{\max})$ (resp. $\varphi \in IUMSO(\Sigma, \mathbb{R}_{\max})$) such that $S = \|\varphi\|$. We denote by ω-*r-Mso*$(\Sigma, \mathbb{R}_{\max}, \Phi)$ (resp. ω-*iu-Mso*$(\Sigma, \mathbb{R}_{\max}, \Phi)$) the class of all RMSO-$\Phi$-definable (resp. IUMSO-$\Phi$-definable) infinitary tree series.

The main result of this section is the subsequent Rabin-type theorem.

Theorem 4. ω-*Rec*$(\Sigma, \mathbb{R}_{\max}, \Phi) = \omega$-*r-Mso*$(\Sigma, \mathbb{R}_{\max}, \Phi) = \omega$-*iu-Mso*$(\Sigma, \mathbb{R}_{\max}, \Phi)$.

First, using induction on the structure of formulas, we state the inclusions ω-*r-Mso*$(\Sigma, \mathbb{R}_{\max}, \Phi) \subseteq \omega$-*Rec*$(\Sigma, \mathbb{R}_{\max}, \Phi)$ and ω-*iu-Mso*$(\Sigma, \mathbb{R}_{\max}, \Phi) \subseteq \omega$-*Rec*$(\Sigma, \mathbb{R}_{\max}, \Phi)$.

Lemma 8. *Let $\varphi, \psi \in MSO(\Sigma, \mathbb{R}_{\max})$. Then*

(i) (cf. [32], Lm. 22) if φ is an atomic formula or the negation of an atomic formula, then $\|\varphi\|$ is an ω-recognizable step function,

(ii) (cf. [32], Lm. 23) if $\|\varphi\|, \|\psi\|$ are (Φ,ω)-recognizable (resp. ω-recognizable step functions), then $\|\varphi \vee \psi\|$ and $\|\varphi \wedge \psi\|$ are (Φ,ω)-recognizable (resp. ω-recognizable step functions),

(iii) (cf. [32], Lm. 24) if $\|\varphi\|$ *is* (Φ,ω)*-recognizable (resp. an* ω*-recognizable step function), then* $\|\exists x \,.\, \varphi\|$ *and* $\|\exists X \,.\, \varphi\|$ *are* (Φ,ω)*-recognizable (resp.* ω*-recognizable step functions),*

(iv) (cf. [32], Lm. 25) if $\|\varphi\|$ *is an* ω*-recognizable step function, then* $\|\forall x \,.\, \varphi\|$ *is* (Φ,ω)*-recognizable, and*

(v) (cf. [32], Lm. 26) if $\|\varphi\| = 0_L$ *where the tree language* $L \subseteq T^{\omega,v}_{\Sigma_\varphi}$ *is* ω*-recognizable, then* $\|\forall X \,.\, \varphi\|$ *is an* ω*-recognizable step function.*

Therefore, we get

Proposition 11.

- $\omega\text{-}r\text{-}Mso(\Sigma,\mathbb{R}_{\max},\Phi) \subseteq \omega\text{-}Rec(\Sigma,\mathbb{R}_{\max},\Phi)$.
- $\omega\text{-}iu\text{-}Mso(\Sigma,\mathbb{R}_{\max},\Phi) \subseteq \omega\text{-}Rec(\Sigma,\mathbb{R}_{\max},\Phi)$.

Conversely, following the proof of Proposition 29 in [32] we state

Proposition 12.

$$\omega\text{-}Rec(\Sigma,\mathbb{R}_{\max},\Phi) \subseteq \omega\text{-}r\text{-}Mso(\Sigma,\mathbb{R}_{\max},\Phi) \cap \omega\text{-}iu\text{-}Mso(\Sigma,\mathbb{R}_{\max},\Phi).$$

Proof of Theorem 4. It is immediate by Propositions 11 and 12. □

Acknowledgment

We are deeply grateful to two anonymous referees for their valuable and extended comments. Especially, one of them helped a lot to develop the presentation of the paper. We are also pleased to thank Heiko Vogler for suggestions to shorten the first and second versions of the paper.

References

[1] A. Alexandrakis, S. Bozapalidis, Weighted grammars and Kleene's theorem, *Inform. Process. Lett.* 24(1)(1987) 1–4.

[2] B. Borchardt, *The Theory of Recognizable Tree Series*, PhD Thesis, Dresden University of Technology, 2004.

[3] B. Borchardt, Code selection for tree series transducers, in: *Proceedings of CIAA'04, Lecture Notes in Comput. Sci.* 3317(2005) 57–67.

[4] G. Bruns, P. Godefroid, Model-checking with multi-valued logics, in: *Proceedings of ICALP 2004, Lecture Notes in Comput. Sci.* 3142(2004) 281–293.

[5] K. Chatterjee, L. Doyen, T.A. Henzinger, Quantitative languages, in: *Proceedings of CSL 2008, Lecture Notes in Comput. Sci.* 5213(2008) 385-400.

[6] K. Chatterjee, L. Doyen, T.A. Henzinger, Composition and alternation for weighted automata, *EPLF Technical Report MTC-REPORT-2008-004.*

[7] M. Chechik, B. Devereux, A. Gurfinkel, Model-checking infinite state-space systems with fine-grained abstractions using SPIN, in: *Proceedings of SPIN 2001, Lecture Notes in Comput. Sci.* 2057(2001) 16–36.

[8] B. Courcelle, Fundamental properties of infinite trees, *Theoret. Comput. Sci.* 25(1983) 95–169.

[9] L. de Alfaro, T.A. Henzinger, R. Majumda, Discounting the future in systems theory, in: *Proceeedings of ICALP 2003, Lecture Notes in Comput. Sci.* 2719(2003) 1022-1037.

[10] J. Doner, Tree acceptors and some of their applications, *J. Comput. System Sci.* 4(1970) 406–451.

[11] M. Droste, P. Gastin, Weighted automata and weighted logics, *Theoret. Comput. Sci.* 380(2007) 69-86; extended abstract in: *Proceedings of ICALP 2005, Lecture Notes in Comput. Sci.* 3580(2005) 513–525.

[12] M. Droste, P. Gastin, On aperiodic and star-free formal power series in partially commuting variables, *Theory Comput. Syst.* 42(2008) 608-631.

[13] M. Droste, P. Gastin, Weighted automata and weighted logics, in: *Handbook of Weighted Automata* (M. Droste, W.Kuich, H.Vogler, eds.) Chap. 5, Springer-Verlag 2009.

[14] M. Droste, D. Kuske, Skew and infinitary formal power series, *Theoret. Comput. Sci.* 366(2006) 189–227; extended abstract in: *Proceedings of ICALP 2003, Lecture Notes in Comput. Sci.* 2719(2003) 426–438.

[15] M. Droste, C. Pech, H. Vogler, A Kleene theorem for weighted tree automata, *Theory Comput. Syst.* 38(2005) 1–38.

[16] M. Droste, G. Rahonis, Weighted automata and weighted logics with discounting, *Theoret. Comput. Sci.* 410(2009) 3481–3494; extended abstract in: *Proceedings of CIAA 2007, Lecture Notes in Comput. Sci.* 4783(2007) 73-84.

[17] M. Droste, J. Sakarovitch, H. Vogler, Weighted automata with discounting, *Inform. Process. Lett.* 108(2008) 23–28.

[18] M. Droste, H. Vogler, Weighted tree automata and weighted logics, *Theoret. Comput. Sci.* 366(2006) 228–247.

[19] M. Droste, H. Vogler, Weighted logics for unranked tree automata, *Theory Comput. Syst.*, to appear.

[20] C. Ferdinand, H. Seidl, R. Wilhelm, Tree automata for code selection, *Acta Inform.* 31(1994) 741–760.

[21] J. Filar, K. Vrieze, *Competitive Marcov Decision Processes,* Springer-Verlag, 1997.

[22] Z. Fülöp, H. Vogler, Weighted Tree Automata and Tree Transducers, in: *Handbook of Weighted Automata* (M. Droste, W.Kuich, H.Vogler, eds.) Chap. 9, Springer-Verlag 2009.

[23] I. Guessarian, Algebraic Semantics, *Lecture Notes in Comput. Sci.* 99(1981).

[24] A. Gurfinkel, M. Chechik, Multi-valued model checking via classical model checking, in: *Proceedings of CONCUR 2003, Lecture Notes in Comput. Sci.* 2761(2003) 266–280.

[25] D. Harel, Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness, *J. ACM* 33(1986) 224–248.

[26] B. Khoussainov, A. Nerode, *Automata Theory and its Applications,* Birkhäuser, Boston, 2001.

[27] W. Kuich, On skew formal power series, in: *Proceedings of the 1st International Conference on Algebraic Informatics,* Thessaloniki 2005, pp.7-30.

[28] E. Mandrali, *Weighted tree automata with discounting,* Master Thesis, Aristotle University of Thessaloniki, 2008, http://users.auth.gr/~elemandr/Mandrali_ms.pdf

[29] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* 141(1969) 1–35.

[30] J.R. Rico-Juan, J. Carela-Rubio, R. Carracso, Stochastic k-testable tree languages, in: *Proceedings of ICGE 2002, Lecture Notes in Artificial Intelligence* 2484(2002) 199–212.

[31] J.R. Rico-Juan, J. Carela-Rubio, R. Carracso, Smoothing and compression with stochastic k-testable tree languages, *Pattern Recognition* 38(2002) 1420–1430.

[32] G. Rahonis, Weighted Muller tree automata and weighted logics, *J. Autom. Lang. Comb.* 12(2007) 455-483.

[33] H. Seidl, Finite tree automata with cost functions, *Theoret. Comput. Sci.* 126(1994) 113-142.

[34] L.S. Shapley, Stochastic games, *Roc. National Acad. of Sciences* 39(1953) 1095-1100.

[35] J.W. Thatcher, J.B. Wright, Generalized finite automata theory with application to a decision problem of second-order logic, *Math. Syst. Theory* 2(1968) 57–81.

[36] W. Thomas, Automata on infinite objects, in: *Handbook of Theoretical Computer Science, vol. B* (J. v. Leeuwen, ed.), Elsevier Science Publishers, Amsterdam 1990, pp. 135–191.

[37] W. Thomas, Languages, automata and logic, in: *Handbook of Formal Languages* vol. 3 (G. Rozenberg, A. Salomaa, eds.), Springer 1997, pp. 389–485.

[38] M.Y. Vardi, An automata-theoretic approach to linear temporal logic, in: *Logics in Concurrency, Lecture Notes in Comput. Sci.* 1043(1996) 238–266.

[39] W. Wechler, *Universal Algebra for Computer Scientists,* EATCS Monographs on Theoretical Computer Science, vol. 25, Springer-Verlag 1992.

Received 17th July 2008

AUTOMATA AND FORMAL LANGUAGES

Guest Editors:

Erzsébet Csuhaj-Varjú

Computer and Automation
Research Institute
Hungarian Academy of Sciences
Budapest, Hungary
csuhaj@sztaki.hu

Zoltán Ésik

Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
ze@inf.u-szeged.hu

Preface

The 12th International Conference on Automata and Formal Languages (AFL 2008) was held in Balatonfüred, May 27–30, 2008, and was organized by the Computer and Automation Research Institute of the Hungarian Academy of Sciences.

The AFL series was initiated by the late Prof. István Peák (1936-1989). He organized the AFL conferences in 1980, 1982, 1984, 1986 and 1988 and started the organization of AFL in 1990. These conferences were all held in the hills around Salgótarján. In 1986 and 1988, the title of the conference was Automata, Languages and Programming Systems.

The scientific program featured invited lectures by Viliam Geffert (Kosice), Markus Lohrey (Leipzig), Ion Petre (Turku) and Jacques Sakarovitch (Paris), and presentations of 24 contributed papers representing 13 countries selected by the international Program Committee.

This special issue contains the full length papers of 7 contributed articles presented at the conference. All papers were refereed in accordance with the journal's standards. We are grateful to the members of the Program Committee of AFL 2008 and their subreferees as well as to all those who served as reviewers of the papers submitted to this special issue.

Budapest and Szeged, October 2009.

Erzsébet Csuhaj-Varjú and Zoltán Ésik
Guest Editors

Acta Cybernetica 19 (2009) 445–464.

Languages Convex with Respect to Binary Relations, and Their Closure Properties*

Thomas Ang† and Janusz Brzozowski†

Abstract

A language is prefix-convex if it satisfies the condition that, if a word w and its prefix u are in the language, then so is every prefix of w that has u as a prefix. Prefix-convex languages include prefix-closed languages at one end of the spectrum, and prefix-free languages, which include prefix codes, at the other. In a similar way, we define suffix-, bifix-, factor-, and subword-convex languages and their closed and free counterparts. This provides a common framework for diverse languages such as codes, factorial languages and ideals. We examine the relationships among these languages. We generalize these notions to arbitrary binary relations on the set of all words over a given alphabet, and study the closure properties of such languages.

Keywords: closed, closure, code, convex, factor, factorial, free, ideal, relation, language, prefix, subword, suffix

1 Introduction

This section introduces our basic terminology and notation, defines the scope of our work, and states some preliminary observations. Previous research is described in Section 2.

A note concerning the terminology is in order. We have used the term *continuous languages* in several publications [1, 5, 6, 7]. However, the term *convex languages* had been used for the same concept much earlier in [20]. Consequently we revert to the earlier terminology here.

Let Σ be an alphabet, and Σ^*, the free monoid generated by Σ, with ϵ as the empty word. A language over an alphabet Σ is any subset of Σ^*. If $L \subseteq \Sigma^*$, the complement of L with respect to Σ^* is denoted by $\overline{L}$. When convenient, we use the customary notation for regular expressions, with $+$ for union, juxtaposition for concatenation, and * for Kleene closure.

*This work was supported by the Natural Sciences and Engineering Research Council of Canada under grant no. OGP0000871.

†David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada N2L 3G1, E-mail: tang@student.cs.uwaterloo.ca,brzozo@uwaterloo.ca

Suppose $\trianglelefteq$ is a binary relation on Σ^*; if $u \trianglelefteq v$ and $u \neq v$, we write $u \triangleleft v$. Let $\trianglerighteq$ be the converse binary relation, that is, let $u \trianglerighteq v$ if and only if $v \trianglelefteq u$. The reflexive-and-transitive closure of $\trianglelefteq$ is denoted by $\trianglelefteq^*$.

Definition 1 and Proposition 1 below are generalizations of some results of Haines [10] and Thierrin [20]. See Section 2 for a further discussion.

Definition 1. *A language L is $\trianglelefteq$-convex if $u \trianglelefteq v$, $u \trianglelefteq w$, and $v \trianglelefteq w$ with $u, w \in L$ imply $v \in L$. It is $\trianglelefteq$-free if $v \triangleleft w$ and $w \in L$ imply $v \notin L$. It is $\trianglelefteq$-closed if $v \trianglelefteq w$ and $w \in L$ imply $v \in L$. It is $\trianglerighteq$-closed if $v \trianglerighteq w$ and $w \in L$ imply $v \in L$.*

For an arbitrary relation $\trianglelefteq$ on Σ^*, let

$$\trianglelefteq L = \{v \in \Sigma^* \mid v \trianglelefteq^* w \text{ for some } w \in L\}$$

and

$$L \trianglelefteq = \{v \in \Sigma^* \mid v \trianglerighteq^* w \text{ for some } w \in L\}.$$

The following are easily verified:

Proposition 1. *Let $\trianglelefteq$ be an arbitrary relation on Σ^*. Then*

1. *A language is $\trianglelefteq$-convex if and only if it is $\trianglerighteq$-convex.*
2. *A language is $\trianglelefteq$-free if and only if it is $\trianglerighteq$-free.*
3. *Every $\trianglelefteq$-closed language and every $\trianglerighteq$-closed language is $\trianglelefteq$-convex.*
4. *A language is $\trianglelefteq$-closed if and only if its complement is $\trianglerighteq$-closed.*
5. *A language L is $\trianglelefteq$-closed ($\trianglerighteq$-closed) if and only if $L = \trianglelefteq L$ ($L = L\trianglelefteq$).*

Example 1. For $w \in \Sigma^*$, let $|w|_2$ be the length of w modulo 2. Let $\Sigma = \{a\}$ and let $\trianglelefteq$ be the binary relation $\leq_2$ defined by

$$u \leq_2 v \text{ if } |u|_2 \leq |v|_2.$$

The $\leq_2$-convex languages are $J = a^*$, $K = a(aa)^*$, $L = (aa)^*$, and $\emptyset$. The $\leq_2$-closed languages are J, L, and $\emptyset$. The $\leq_2$-free languages are $\emptyset$ and all the singleton languages $\{w\}$, for $w \in \Sigma^*$. Note that there are $\leq_2$-free languages that are not $\leq_2$-convex. For instance, $\{aa\}$ is $\leq_2$-free but not $\leq_2$-convex, because $aa \leq_2 \varepsilon$, $aa \leq_2 aa$, $\varepsilon \leq_2 aa$, but $\varepsilon \notin L$.

Proposition 2. *If $\trianglelefteq$ is antisymmetric, then every $\trianglelefteq$-free language is $\trianglelefteq$-convex. If $\trianglelefteq$ is reflexive and every $\trianglelefteq$-free language is $\trianglelefteq$-convex then $\trianglelefteq$ is antisymmetric.*

Proof. Suppose L is $\trianglelefteq$-free and $\trianglelefteq$ is antisymmetric. If L is not $\trianglelefteq$-convex, then there exist $u, w \in L$, $v \notin L$, such that $u \trianglelefteq v$, $u \trianglelefteq w$, and $v \trianglelefteq w$. Thus $u \trianglelefteq v$ and $v \trianglelefteq w$ and, since $\trianglelefteq$ is antisymmetric, we have $u \neq w$. Thus we have $u, w \in L$ and $u \triangleleft w$, contradicting that L is $\trianglelefteq$-free.

Conversely, suppose $\trianglelefteq$ is reflexive and every $\trianglelefteq$-free language is $\trianglelefteq$-convex, but $\trianglelefteq$ is not antisymmetric. Then there exist $v, w \in \Sigma^*$ such that $w \trianglelefteq v$, $v \trianglelefteq w$ and $v \neq w$. Since $\trianglelefteq$ is reflexive, we have $w \trianglelefteq w$. The language $\{w\}$ is $\trianglelefteq$-free but not $\trianglelefteq$-convex. Note that, if reflexivity is absent, v and w do not violate convexity. □

Usually in our applications we deal with partial order relations, so reflexivity and antisymmetry hold. If the binary relation is understood, we call a language *convex, free, closed,* or *converse closed.*

If $u, v, w \in \Sigma^*$ and $w = uv$, then u is a *prefix* of w and v is a *suffix* of w. If v is a prefix of w, we write $v \leq w$; if also $v \neq w$, then $v < w$. If v is a suffix of w, we write $v \preceq w$; if also $v \neq w$, then $v \prec w$. If $w = xvy$ for some $v, x, y \in \Sigma^*$, then v is a *factor* of w. Note that a prefix or suffix of w is also a factor of w. If v is a factor of w, we write $v \sqsubseteq w$; if also $v \neq w$, then $v \sqsubset w$. If $w = w_0 a_1 w_1 \cdots a_n w_n$, where $a_1, \ldots, a_n \in \Sigma$, and $w_0, \ldots, w_n \in \Sigma^*$, then $v = a_1 \cdots a_n$ is a *subword* of w; note that every factor of w is a subword[1] of w. If v is a subword of w, we write $v \Subset w$; if also $v \neq w$, then $v \subset w$. The relations $\leq$, $\preceq$, $\sqsubseteq$, and $\Subset$ are partial orders on Σ^*.

We apply Definition 1 to the following special cases:

$\trianglelefteq$ is $\leq$: If we use the relation "is a prefix of", then we get prefix-convex languages [6]. Prefix-free languages, except $\{\varepsilon\}$, are prefix codes [4], prefix-closed languages are complements of right ideals, and converse closed languages are the right ideals, that is, have the form $L\Sigma^*$, $L \subseteq \Sigma^*$. See Proposition 7.

$\trianglelefteq$ is $\preceq$: If we use the relation "is a suffix of", then we get the suffix-convex languages. Suffix-free languages, except $\{\varepsilon\}$, are suffix codes [4], suffix-closed languages are complements of left ideals, and converse closed languages are the left ideals, that is, have the form $\Sigma^* L$. See Proposition 7.

$\trianglelefteq$ is $\sqsubseteq$: If we use the relation "is a factor of"[2], we get factor-convex languages. Factor-free languages, except $\{\varepsilon\}$, are infix codes [19], factor-closed languages are factorial languages [15], which are complements of two-sided ideals, and converse closed languages are the ideals, that is, have the form $\Sigma^* L \Sigma^*$. See Proposition 6.

$\trianglelefteq$ is $\Subset$: If we use the relation "is a subword of"[3], we get subword-convex languages. Subword-free languages, except $\{\varepsilon\}$, are hypercodes [19], subword-closed languages are of the form $K = \overline{L} = \bigcup_{a_1 \cdots a_i \in L} \Sigma^* a_1 \Sigma^* \cdots a_i \Sigma^*$, and converse closed languages are of the form L above. See Section 2.

If a language is both prefix- and suffix-convex it is *bifix-convex*[4]. If it is both prefix- and suffix-free it is *bifix-free*; if it is not $\{\varepsilon\}$, it is then a bifix code [4]. If it is both prefix- and suffix-closed, it is *bifix-closed.* Note that bifix-closed and bifix-free languages can be defined as $(\leq \cup \preceq)$-closed and $(\leq \cup \preceq)$-free languages, respectively, but bifix-convex languages cannot be derived from a single relation.

The remainder of the paper is structured as follows. Previous work on convex languages is described in Section 2. In Section 3 we show the relations among the

[1] The word "subword" is often used to mean "factor"; here by a "subword" we mean a subsequence.

[2] This relation is called the "infix order" in [19].

[3] This relation is called the "embedding order" in [10].

[4] The word "bifix" is sometimes used to describe a word that is both a prefix and a suffix. Here we follow [12, 18]. The term "biprefix" is used in [4].

prefix- and suffix-convex classes of languages and their subclasses. In Section 4 we study the closure properties of the X-convex, X-closed and X-free classes of languages, where X stands for prefix, suffix, bifix, factor or subword. The converse X-closed classes are considered in Section 5. Special properties of closure under concatenation and star are studied in Section 6, and Section 7 concludes the paper.

2 Previous results and generalizations

For consistency, we use our notation and terminology when discussing previous work, but some of the key original terms are also mentioned.

In 1969 Haines proved the following results [10]:

Theorem 1 (Haines). *Every subword-free language is finite.*

He called the subword relation *embedding*. He also defined (what we call) the *subword closure* of any language $L \subseteq \Sigma^*$ which is the set of all words that are subwords of words in L:

$$_{\Subset}L = \{u \in \Sigma^* \mid u \text{ is a subword of } v \text{ for some } v \in L\}.$$

Dually, he defined (what we call) the *converse subword closure* of any language $L \subseteq \Sigma^*$ which is the set of all words that contain a word of L as a subword:

$$L_{\Subset} = \{v \in \Sigma^* \mid u \text{ is a subword of } v \text{ for some } u \in L\}.$$

Theorem 2 (Haines). *For any $L \subseteq \Sigma^*$, there exist finite languages F and G, such that*

$$_{\Subset}L = \overline{F_{\Subset}} = \overline{\bigcup_{a_1 \cdots a_i \in F} \Sigma^* a_1 \Sigma^* \cdots a_i \Sigma^*},$$

and

$$L_{\Subset} = G_{\Subset} = \bigcup_{a_1 \cdots a_i \in G} \Sigma^* a_1 \Sigma^* \cdots a_i \Sigma^*.$$

Theorem 3 (Haines). *The languages $_{\Subset}L$ and $L_{\Subset}$ are regular, for every $L \subseteq \Sigma^*$.*

Haines noted that Theorem 1 is false for the factor relation, because $L = \{ab^n a \mid n \geq 1\}$ is an infinite factor-free language. It is also false for the prefix and suffix relations, since L is an infinite prefix- and suffix-free language.

For a discussion of earlier work related to the results of Haines see the paper by Kruskal [14].

In 1973 Thierrin introduced convex languages for the subword partial order [20]. He called a language *convex* if it is $\Subset$-convex, *left convex* if it is $\Subset$-closed, and *right-convex* if it is $\Supset$-closed. He also defined a language to be *strongly convex* if it is closed under nonempty subwords, that is, if $v \neq \varepsilon$, $v \Subset w$, and $w \in L$ implies $v \in L$. This last concept is outside the scope of this work; we refer the reader to [20].

Proposition 3 (Thierrin). *A language is $\Subset$-convex if and only if it is an intersection of a $\Subset$-closed and a $\Supset$-closed language. Equivalently, a language L is $\Subset$-convex if and only if there exist $\Subset$-closed languages M and N such that $L = M \setminus N$.*

Corollary 1 (Thierrin). *Every $\Subset$-convex language is regular.*

Proposition 3 can be generalized with an added condition on $\trianglelefteq$.

Proposition 4. *If there exist $M, N \subseteq \Sigma^*$ such that M and N are $\trianglelefteq$-closed and $L = M \setminus N$, then L is $\trianglelefteq$-convex. If $\trianglelefteq$ is transitive and L is $\trianglelefteq$-convex, then there exist $M, N \subseteq \Sigma^*$ such that M and N are $\trianglelefteq$-closed and $L = M \setminus N$.*

Proof. Suppose $L = M \setminus N$, where M and N are $\trianglelefteq$-closed, and L is not $\trianglelefteq$-convex. Then there exists a triple $(u \in L, v \notin L, w \in L)$, such that $u \trianglelefteq v$, $u \trianglelefteq w$, and $v \trianglelefteq w$. We must have $u, w \in M$, and $u, w \notin N$, that is, $u, w \in \overline{N}$. If $v \in M$, then also $v \in N$ and $v \notin \overline{N}$. This means that $\overline{N}$ is not $\trianglelefteq$-convex. But, $\overline{N}$ is $\trianglerighteq$-closed by Proposition 1 (4), and every $\trianglerighteq$-closed language is $\trianglelefteq$-convex by Proposition 1 (3), which is a contradiction. Hence we must have $v \notin M$. But this now means that M is not $\trianglelefteq$-convex, and hence cannot be $\trianglelefteq$-closed—again a contradiction.

Conversely, suppose that $\trianglelefteq$ is transitive, and L is $\trianglelefteq$-convex. Let $M = {\trianglelefteq}L$; then M is $\trianglelefteq$-closed by definition. Let

$$N = {\trianglelefteq}L \setminus L = \{v \in \Sigma^* \mid v \notin L \text{ and } v \trianglelefteq^* w \text{ for some } w \in L\}.$$

Since $\trianglelefteq$ is transitive, we also have

$$N = \{v \in \Sigma^* \mid v \notin L \text{ and } v \trianglelefteq w \text{ for some } w \in L\}.$$

We claim that N is also $\trianglelefteq$-closed. For suppose $u \trianglelefteq v$ for some $v \in N$ such that $v \trianglelefteq w$, for some $w \in L$. Then $u \trianglelefteq w$ by transitivity. If $u \in L$, then L cannot be $\trianglelefteq$-convex because of the triple $(u \in L, v \notin L, w \in L)$. Hence we must have $u \notin L$, and N is $\trianglelefteq$-closed. Now $M \setminus N = M \cap \overline{N} = {\trianglelefteq}L \cap (\overline{{\trianglelefteq}L} \cup L) = {\trianglelefteq}L \cap L = L$. □

Example 2. Let $\Sigma = \{a\}$, and let $\trianglelefteq = \{(\varepsilon, a), (a, aa)\}$; then $\trianglelefteq$ is not transitive since (ε, aa) is not in the relation. If $L = \{\varepsilon, aa\}$, then L is $\trianglelefteq$-convex, but not $\trianglelefteq$-closed, since $a \trianglelefteq aa$, $aa \in L$ and $a \notin L$. Suppose L can be expressed as $L = M \setminus N$, where both M and N are $\trianglelefteq$-closed. Then M must contain L and be closed; hence $a \in M$. Now N must contain a; otherwise $a \in M \setminus N$, and $M \setminus N \neq L$. However, since N must be closed, it must contain ε, since $\varepsilon \trianglelefteq a$. But then $M \setminus N$ does not contain ε, which is a contradiction. Therefore, Proposition 3 does not hold here.

On the other hand, lack of transitivity does not prevent *all* languages from satisfying Proposition 3. For example, the language $K = \{a\}$ is $\trianglelefteq$-convex and not $\trianglelefteq$-closed, but can be expressed as $K = \{\varepsilon, a\} \setminus \{\varepsilon\}$, which is a difference between two $\trianglelefteq$-closed languages.

Proposition 5 (Thierrin). *If L is a $\Subset$-closed or a $\Supset$-closed language over Σ, then the syntactic monoid M_L is finite and contains a disjunctive zero z such that $ab = z$, $a, b \in M_L$, implies $axb = z$ for every $x \in M_L$.*

A language L is *noncounting* [20] if there exists an integer $k \geq 0$ such that, for arbitrary $u, v, w \in \Sigma^*$, $uv^k w \in L$ if and only if $uv^{k+1}w \in L$.

Corollary 2 (Thierrin). *Every $\Subset$-convex, $\Subset$-closed, and $\Supset$-closed language is a noncounting regular language, and hence, a star-free language.*

Corollary 2 does not hold for the prefix, suffix, and factor relations. For example, $K = (aa)^*b\Sigma^*$, $L = \Sigma^*b(aa)^*$ and $M = \Sigma^*b(aa)^*b\Sigma^*$ are converse prefix-closed, converse suffix-closed and converse factor-closed, respectively, but they are not noncounting. However, convex languages with respect to these three relations are noncounting in the case of the one-letter alphabet.

Properties of $\Subset$-free languages were studied by Shyr and Thierrin [19] under the name of *hypercodes*. There is an extensive literature on codes characterized as antichains with respect to binary relations in free monoids. For example, languages that are both factor-free codes and $\Subset$-convex are studied in [9]. See also [11, 13, 18] and the references contained therein for further examples. It is not our purpose in this paper to deal with this topic in depth, but only to point out how various classes of these languages fit into the framework of convex languages, and to study the closure properties of convex languages.

In 1990, de Luca and Varricchio characterized factor-closed languages, which they called *factorial:*

Proposition 6 (De Luca & Varricchio). *A language L is factorial (that is, $\sqsubseteq$-closed) if and only if it is the complement of a two-sided ideal, that is, if and only if $L = \overline{\Sigma^* K \Sigma^*}$, for some language K.*

In Proposition 6, K can be taken to be regular if L is regular.

We have analogous results for prefix-closed and suffix-closed languages:

Proposition 7. *A language L is prefix-closed (suffix-closed) if and only if it is the complement of a right (left) ideal, that is, if and only if $L = \overline{K\Sigma^*}$, ($L = \overline{\Sigma^* K}$) for some language K. Moreover, K can be taken to be regular if L is regular.*

Proof. The proof parallels the proof of Proposition 6 in [15]. Let $\leq L$ be the set of all prefixes of words in L; thus, if L is prefix-closed, then $L = \leq L$. Now let $K = \overline{\leq L}$. One verifies that $u \in K$ implies $uv \in K$ for all $v \in \Sigma^*$, that is, $K = K\Sigma^*$, and $L = \leq L = \overline{K} = \overline{K\Sigma^*}$. Note that K is regular if L is regular. Conversely, suppose $L = \overline{K\Sigma^*}$ for some K, $w = uv \in L$, and $u \notin L$. Then $u \in K\Sigma^*$, $u = u'u''$, for some $u' \in K$, $u'' \in \Sigma^*$; and $w = u'u''v$ must also be in $K\Sigma^*$, which is a contradiction. Thus L is prefix-closed.

A dual argument proves the result for suffix-closed languages. □

Prefix-convex languages were studied in connection with trace-assertion specifications [6, 7] (under the name of prefix-continuous languages). Here a software module is modeled by an automaton in which the states are represented by words over the input alphabet. It was shown in [6], for deterministic automata, that

the automaton is well-behaved if the set of words representing the states is prefix-convex. This result was extended to nondeterministic automata in [5]. Applications of these methods to the specification of software modules were discussed in [7].

Closure properties studied by Thierrin [20] are discussed in later sections.

3 Examples of convex languages

For convenience, we first consider $\trianglelefteq$-convex, $\trianglelefteq$-free, and $\trianglelefteq$-closed languages, where $\trianglelefteq$ ranges over $\{\leq, \preceq, \sqsubseteq, \Subset\}$. If a nonempty language is prefix-convex (respectively, suffix-, bifix-, factor-, or subword-convex), then it is prefix-closed (respectively, suffix-, bifix-, factor-, or subword-closed) if and only if it contains ε. The empty language $\emptyset$ and the language $\{\varepsilon\}$ vacuously satisfy the $\trianglelefteq$-convex, $\trianglelefteq$-free, and $\trianglelefteq$-closed conditions if $\trianglelefteq \in \{\leq, \preceq, \sqsubseteq, \Subset\}$. Also, since ε is a prefix, suffix, factor, and subword of every word, $\emptyset$ and $\{\varepsilon\}$ are the only two languages that are both $\trianglelefteq$-free and $\trianglelefteq$-closed.

Factorial languages are defined as factor-closed languages, for example, in [2, 15], and as bifix-closed languages, for example, in [16]. This is justified in view of the following:

Remark 1. A language is factor-closed if and only if it is bifix-closed.

Proof. If L is factor-closed, then it is also bifix-closed, since every prefix and suffix is also a factor. Conversely, let L be a bifix-closed language and let $w \in L$. Suppose v is any factor of $w = xvy$; then $xv \in L$ since xv is a prefix of w, and $v \in L$ because v is a suffix of xv. Therefore L is factor-closed. □

Factorial languages have received considerable attention. For example, their decompositions are studied in [2], their combinatorial properties in [15], and their complexity issues in [17]. We return to these languages later.

Figure 1 shows the various classes of languages partially ordered under set containment, where P, S, B, F, and W, stand for prefix, suffix, bifix, factor, and subword, respectively, PC, PF and PCL stand for prefix-convex, prefix-free, and prefix-closed languages, *etc.* The classes in small rectangular boxes are closed under concatenation; we discuss this later. The classes in the large rectangle correspond to codes. The only difference between the solid and dashed lines is that the dashed lines indicate free and closed languages as special cases of convex languages, while solid lines show classes defined by changing the underlying binary relation.

Proposition 8. *All containments shown in Figure 1 are proper, and there are no other containments, except those implied by transitivity.*

Proof. First, we verify that the containments shown do indeed hold. Any class of the form BX, where $X \in \{C, CL, F\}$ is the intersection of PX and SX, by definition. Also, $BX \supseteq FX$, because every prefix and suffix is a factor, and $FX \supseteq WX$, because every factor is a subword. This explains the solid lines. Next, for $Y \in$

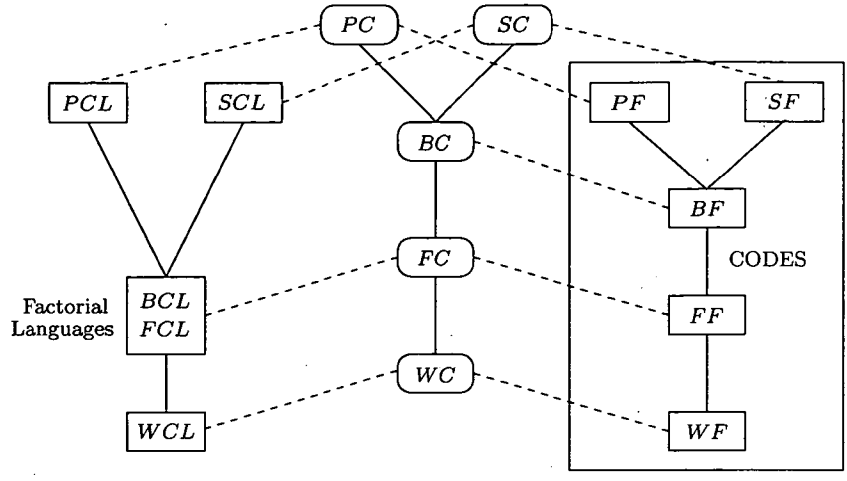


Figure 1: Classes of convex languages

$\{P, S, B, F, W\}$, classes YCL and YF are special cases of YC by Propositions 1(3) and 2; this accounts for the dashed lines.

Second, we show that no class contains any other class except as shown, or implied by transitivity of set containment. We consider each class in turn, starting with the maximal ones.

The prefix-convex class PC: It suffices to show that PC contains neither SCL nor SF. We have $L_1 = \{\varepsilon, a, ba\} \in SCL \setminus PC$, and $L_2 = \{a, abb\} \in SF \setminus PC$.

The suffix-convex class SC: Use left-right symmetry with PC.

The prefix-closed class PCL: It suffices to show that PCL contains neither SCL nor WF. Since PC does not contain SCL, neither does its subclass PCL. Also, $L_3 = \{a, b\} \in WF \setminus PCL$.

The suffix-closed class SCL: Use left-right symmetry with PCL.

The prefix-free class PF: It suffices to show that PF contains neither SF nor WCL. Since PC does not contain SF, neither does PF. Also, $L_4 = \{\varepsilon, a\} \in WCL \setminus PF$.

The suffix-free class SF: Use left-right symmetry with PF.

The bifix-convex class BC: It suffices to show that BC does not contain any class from $\{PCL, SCL, PF, SF\}$. This follows because $L_5 = \{\varepsilon, a, ab\} \in PCL \setminus BC$, $L_1 = \{\varepsilon, a, ba\} \in SCL \setminus BC$, $L_6 = \{b, aab\} \in PF \setminus BC$, and $L_2 = \{a, abb\} \in SF \setminus BC$.

The bifix-free class BF: It suffices to show that BF does not contain any class from $\{WCL, PF, SF\}$. We have $L_4 = \{\varepsilon, a\} \in WCL \setminus BF$, $L_6 = \{b, aab\} \in PF \setminus BF$, and $L_2 = \{a, abb\} \in SF \setminus BF$.

The factor-convex class FC: It suffices to show that FC does not contain any class from $\{PCL, SCL, BF\}$. Since BC does not contain PCL or SCL, neither does FC. Also, $L_7 = \{b, aba\} \in BF \setminus FC$.

The bifix-closed class BCL: It suffices to show that BCL does not contain any class from $\{PCL, SCL, WF\}$. Since FC does not contain PCL or SCL, neither does BCL. Also, $L_8 = \{a\} \in WF \setminus BCL$.

The factor-free class FF: It suffices to show that FF does not contain any class from $\{WCL, BF\}$. Since BF does not contain WCL, neither does FF. Since FC does not contain BF, neither does FF.

The subword-convex class WC: It suffices to show that WC contains neither BCL nor FF. We have $L_9 = \{\varepsilon, a, b, ab, ba, aba\} \in BCL \setminus WC$, and $L_{10} = \{aa, abba\} \in FF \setminus WC$.

The subword-closed class WCL: It suffices to show that WCL contains neither BCL nor WF. Since WC does not contain BCL, neither does WCL. Also, $L_8 = \{a\} \in WF \setminus WCL$.

The subword-free class WF: It is enough to show that WF contains neither WCL nor FF. Since FF does not contain WCL, neither does WF. Also, $L_{10} = \{aa, abba\} \in FF \setminus WF$.

This completes the proof. □

Remark 2. $PC \cap SCL = PCL \cap SCL = BCL = SC \cap PCL$.

Proof. By definition, $BCL = PCL \cap SCL$. From Figure 1, we have $PC \cap SCL \supseteq BCL$. Conversely, if L is suffix-closed, then it contains ε, which is also a prefix of every word; thus, if L is also prefix-convex, then it is prefix-closed, and hence bifix-closed. The last equality follows by left-right symmetry. □

3.1 One-letter alphabets

The length of a word $w \in \Sigma^*$ is $|w|$, and w^R is the reverse of w. The reverse of L is $L^R = \{w^R \mid w \in L\}$.

Languages over one-letter alphabets have very special properties. Note that, if $L \subseteq \{a\}^*$, then $L = L^R$. Also, the statements "u is a prefix of w", "u is a suffix of w", "u is a factor of w", and "u is a subword of w" are all equivalent to each other and to "$|u| \leq |w|$". Thus the following are easily verified:

Proposition 9. *If $\Sigma = \{a\}$, and $L \subseteq \Sigma^*$, then the following hold:*

1. *If X stands for "prefix", "suffix", "bifix", "factor", or "subword", then all the statements of the form X-convex are equivalent, all the statements of the form X-free are equivalent, and all the statements of the form X-closed are equivalent.*
2. *L is prefix-convex if and only if it is empty, or has the form $\{a^i \mid m \leq i \leq m+n\}$, or $\{a^i \mid m \leq i\} = a^m a^*$, for some $m \geq 0, n \geq 0$.*
3. *L is prefix-closed if and only if it is empty, or has the form $\{a^i \mid 0 \leq i \leq m\}$, for some $m \geq 0$, or $\{a^i \mid 0 \leq i\} = a^*$.*
4. *L is prefix-free if and only if it is empty, or contains only one word.*
5. *If $K, L \subseteq \Sigma^*$ are prefix-convex, then so is KL.*

4 Closure in $\trianglelefteq$-convex languages

Thierrin [20] proved the closure results of Table 1 for subword-convex languages.

Table 1: Thierrin's closure results for the subword relation.

	convex	closed	converse closed
intersection	yes	yes	yes
union	no	yes	yes
complement	no	no	no
concatenation	no	yes	yes
star	no	yes	no

We generalize and extend these results. We first consider the closure properties of convex, free, and closed classes of languages. Converse closed classes are studied in Section 5.

4.1 Intersection, union and complement

Proposition 10. *If $K, L \subseteq \Sigma^*$ are $\trianglelefteq$-convex ($\trianglelefteq$-free, or $\trianglelefteq$-closed), then so is $M = K \cap L$.*

Proof. If M is not $\trianglelefteq$-convex, there exist $u, w \in M$ and $v \notin M$ such that $u \trianglelefteq v$, $u \trianglelefteq w$, and $v \trianglelefteq w$. Since $u, w \in K$ and $u, w \in L$, and K and L are $\trianglelefteq$-convex, we have $v \in K$ and $v \in L$, which contradicts that $v \notin M$.

If M is not $\trianglelefteq$-free, there exist $v, w \in M$ such that $v \triangleleft w$. Since $v, w \in K$, this contradicts that K is $\trianglelefteq$-free.

If M is not $\trianglelefteq$-closed, there exist $w \in M$, $v \notin M$ such that $v \trianglelefteq w$. Then either $v \notin K$ or $v \notin L$. In the first case, $w \in K$ and $v \notin K$ contradicts that K is $\trianglelefteq$-closed. In the second case, L cannot be $\trianglelefteq$-closed. □

Corollary 3. *All the classes in Figure 1 are closed under intersection.*

The following is easily verified:

Proposition 11. *If $K, L \subseteq \Sigma^*$ are $\trianglelefteq$-closed, then so is $K \cup L$. If L^n is $\trianglelefteq$-closed for $n \geq 1$, then so is $\bigcup_{n=1}^{\infty} L_n$.*

Corollary 4. *All the closed classes, PCL, SCL, $BCL = FCL$, and WCL, are closed under union.*

Remark 3. The remaining classes in Figure 1 are not closed under union. Let $K = \{\varepsilon\}$, $L = \{aa\}$; both languages are X-convex and X-free for all $X \in \{P, S, B, F, W\}$. However, $K \cup L$ is neither X-convex nor X-free.

Remark 4. None of the classes is closed under complementation. The language $L = \{a\}$ is in XC for all $X \in \{P, S, B, F, W\}$, but its complement is not. Also, L is in XF, but $\overline{L}$ is not. The language $K = \{\varepsilon\}$ is in XCL, but $\overline{K}$ is not.

4.2 Concatenation and star

In general, convex languages are not closed under concatenation and star, even if the relation $\trianglelefteq$ is one of prefix, suffix, factor or subword relations.

Remark 5. If $L \subseteq \Sigma^*$ is prefix-(suffix-, bifix-, factor-, or subword-)convex, then L^2 is not necessarily prefix-(suffix-, bifix-, factor-, or subword-)convex. Hence these languages are not closed under concatenation.

Proof. $L = \{a, b, ab, ad, ca\}$ is prefix-, suffix-, factor-, and subword-convex, but L^2 is not, since $ab, abca \in L^2$ but $abc \notin L^2$, and $ab, adab \in L^2$, but $dab \notin L^2$. □

For concatenation of converse closed languages, see Section 5. For closed and free languages see Section 6.

Remark 6. If $L \subseteq \Sigma^*$ is prefix-(suffix-, bifix-, factor-, or subword-)convex, then L^* is not necessarily prefix-(suffix-, bifix-, factor-, or subword-)convex. The same holds if we replace "convex" by "free" or "converse closed".

Proof. If $\Sigma = \{a\}$, $L = \{aa\}$ is prefix-, suffix-, bifix-, factor-, and subword-convex, and -free, but $(aa)^*$ is not. Also, $L = aaa^*$ is converse X-closed for all X, but $L^* = L \cup \{\varepsilon\}$ is not. □

For the star of closed languages see Section 6.2.

4.3 Quotients

If $x \in \Sigma^*$ and $L \subseteq \Sigma^*$, then the *left quotient* of L by x is $x^{-1}L = \{w \in \Sigma^* \mid xw \in L\}$. The *right quotient* of L by x is $Lx^{-1} = \{w \in \Sigma^* \mid wx \in L\}$.

A binary relation is *left-invariant* (*right-invariant*) if $u \trianglelefteq v$ implies $xu \trianglelefteq xv$ ($ux \trianglelefteq vx$).[5]

[5]The terms 'left compatible' and 'right compatible' are used in [13, 18].

Proposition 12. *If $\trianglelefteq$ is left-invariant, and L is $\trianglelefteq$-convex ($\trianglelefteq$-free or $\trianglelefteq$-closed), then $M = x^{-1}L$ is $\trianglelefteq$-convex ($\trianglelefteq$-free or $\trianglelefteq$-closed), for any $x \in \Sigma^*$. The same holds if 'left' is replaced by 'right' and '$x^{-1}L$' by 'Lx^{-1}'.*

Proof. Suppose L is $\trianglelefteq$-convex. If M is not $\trianglelefteq$-convex, then there exist $u, w \in M$ and $v \notin M$ such that $u \trianglelefteq v$, $u \trianglelefteq w$, and $v \trianglelefteq w$. If $\trianglelefteq$ is left-invariant, then $xu \trianglelefteq xv$, $xu \trianglelefteq xw$, and $xv \trianglelefteq xw$, and xu and $xw \in L$, while $xv \notin L$. This contradicts that L is $\trianglelefteq$-convex.

Suppose L is $\trianglelefteq$-free. If M is not $\trianglelefteq$-free, there exist $v, w \in M$ such that $v \triangleleft w$; then $xv, xw \in L$. If $\trianglelefteq$ is left-invariant, then $xv \triangleleft xw$, which contradicts that L is $\trianglelefteq$-free.

Suppose L is $\trianglelefteq$-closed. If M is not $\trianglelefteq$-closed, there exist $w \in M$, $v \notin M$ such that $v \trianglelefteq w$; then $xw \in L$ and $xv \notin L$. If $\trianglelefteq$ is left-invariant, then $xv \trianglelefteq xw$, which contradicts that L is $\trianglelefteq$-closed.

The claim for the case where $\trianglelefteq$ is right-invariant follows by duality. □

Example 1 shows that the invariance conditions of Proposition 12 are not necessary. The relation $\leq_2$ is not left-invariant, since $aa \leq_2 a$ but $a(aa) \not\leq_2 a(a)$. The left (and right) quotient of J by any word is J. The left quotient of K by w is K (respectively L), if w has even (respectively odd) length. Similarly, the left quotient of L by w is L (respectively K), if w has even (respectively odd) length. The left quotient of $\emptyset$ is $\emptyset$. Thus the left quotient of every $\leq_2$-convex language is $\leq_2$-convex. Similarly, the left quotient of every $\leq_2$-free language is $\leq_2$-free. On the other hand, the left quotient of L by a is K, which is not $\leq_2$-closed.

Corollary 5. *The classes PC, PCL and PF are closed under left quotient, SC, SCL and SF are closed under right quotient, and WC, WCL and WF are closed under both quotients.*

Remark 7. The classes BC, BF, FC, FCL and FF are not closed under either type of quotient. For let $L = \{\varepsilon, a, b, ab, ba, aba\}$; then L is bifix-convex, factor-convex and factor-closed, but $a^{-1}L = \{\varepsilon, b, ba\}$ and $La^{-1} = \{\varepsilon, b, ab\}$ are not. Also, $L = \{bb, bab\}$ is bifix-free and factor-free, but $b^{-1}L = \{b, ab\}$ and $Lb^{-1} = \{b, ba\}$ are neither.

4.4 Homomorphism and inverse homomorphism

If S is a set, then 2^S is the set of all subsets of S. Let Σ and Δ be alphabets. A *homomorphism* is a map $h : \Sigma^* \to \Delta^*$ such that $h(uv) = h(u)h(v)$ for all $u, v \in \Sigma^*$. If $L \subseteq \Sigma^*$, then $h(L) = \bigcup_{w \in L}\{h(w)\}$. The *inverse homomorphism* of h is $h^{-1} : h(\Sigma^*) \to 2^{\Sigma^*}$ defined by $h^{-1}(x) = \{w \in \Sigma^* \mid h(w) = x\}$, for all $x \in h(\Sigma^*)$. If $L \subseteq h(\Sigma^*)$, then the inverse image of L under h is $h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$. A *substitution* is a map $s : \Sigma^* \to 2^{\Delta^*}$ such that $s(\varepsilon) = \{\varepsilon\}$, $s(uv) = s(u)s(v)$ for all $u, v \in \Sigma^*$, and $s(L) = \bigcup_{w \in L}\{s(w)\}$.

Remark 8. None of the classes from Figure 1 is closed under homomorphism. If $\Sigma = \Delta = \{a\}$, $h(a) = aa$, $L = \{\varepsilon, a\}$, then $h(L) = \{\varepsilon, aa\}$, L is in XC and in XCL,

for all $X \in \{P, S, B, F, W\}$, but $h(L)$ is not. Also, if $L = \{a, b\}$, $h(a) = \varepsilon$, $h(b) = a$, then $h(L) = \{\varepsilon, a\}$. Now L is in XF, but $h(L)$ is not. It follows that none of the classes from Figure 1 is closed under substitution.

Let $\trianglelefteq$ be a binary relation on Σ^*, and $\trianglelefteq'$, a binary relation on Δ^*. Then h is a $(\trianglelefteq, \trianglelefteq')$-*homomorphism*[6] if $u \trianglelefteq v$ implies $h(u) \trianglelefteq' h(v)$.

Proposition 13. *Let $(\Sigma^*, \trianglelefteq)$ and $(\Delta^*, \trianglelefteq')$ be free monoids with binary relations, let $h : \Sigma^* \to \Delta^*$ be a $(\trianglelefteq, \trianglelefteq')$-homomorphism, and let $K \subseteq h(\Sigma^*)$. If K is $\trianglelefteq'$-convex ($\trianglelefteq'$-free, or $\trianglelefteq'$-closed), then $L = h^{-1}(K)$ is $\trianglelefteq$-convex ($\trianglelefteq$-free, or $\trianglelefteq$-closed).*

Proof. Suppose K is $\trianglelefteq'$-convex, but L is not $\trianglelefteq$-convex. Then there exist $u, w \in L$, $v \notin L$ such that $u \trianglelefteq v$, $u \trianglelefteq w$, and $v \trianglelefteq w$. Since h is a $(\trianglelefteq, \trianglelefteq')$-homomorphism, we also have $h(u), h(w) \in K$, $h(v) \notin K$, and $h(u) \trianglelefteq' h(v)$, $h(u) \trianglelefteq' h(w)$, and $h(v) \trianglelefteq' h(w)$, which contradicts that K is $\trianglelefteq'$-convex.

Suppose K is $\trianglelefteq'$-free, but $L = h^{-1}(K)$ is not $\trianglelefteq$-free. Then there exist $v, w \in L$ such that $v \triangleleft w$. Since h is a $(\trianglelefteq, \trianglelefteq')$-homomorphism, we also have $h(v) \triangleleft' h(w)$, which contradicts that K is $\trianglelefteq'$-free.

Suppose K is $\trianglelefteq'$-closed, but $L = h^{-1}(K)$ is not $\trianglelefteq$-closed. Then there exist $w \in L$, $v \notin L$ such that $v \trianglelefteq w$. If h is a $(\trianglelefteq, \trianglelefteq')$-homomorphism, then $h(w) \in K$, $h(v) \notin K$, and $h(v) \trianglelefteq' h(w)$, which contradicts that K is $\trianglelefteq'$-closed. □

Corollary 6. *All the classes in Figure 1 are closed under inverse homomorphism.*

Proof. If u is a prefix (suffix, factor, or subword) of v and h is a homomorphism, then $h(u)$ is a prefix (suffix, factor, or subword) of $h(v)$. Thus, we have a $(\trianglelefteq, \trianglelefteq')$-homomorphism for all $\trianglelefteq \in \{\leq, \preceq, \sqsubseteq, \Subset\}$. □

5 Converse closed languages

For $X \in \{P, S, F, W\}$, let XCC be the class of converse closed languages corresponding to the prefix, suffix, factor, and subword relations, respectively. By Proposition 1 (3), all these languages are convex. Similarly, let XC represent the convex classes and XCL, the closed classes.

The classes XCC in Figure 2 are the converse closed classes, which are shown in double rectangles. (We explain TR and TR' later.) Each converse closed class $XCC = \{\overline{L} \mid L \in XCL\}$ is in 1-1 correspondence with the corresponding closed class. Note that each class XC contains languages that are not in $XCL \cup XCC \cup XF$. For example, $\{a, aa\}$ is in XC but it is not in $XCL \cup XCC \cup XF$, for all $X \in \{P, S, F, W\}$.

Applying Propositions 10, 11, 12, and 13 for intersection, union, quotient, and inverse homomorphism, respectively, to the relation $\trianglerighteq$, we obtain:

Corollary 7. *All the classes of the form XCC are closed under intersection, union, and inverse homomorphism. Moreover, PCC is closed under left quotient, SCC, under right quotient, and WCC, under both.*

[6] In the terminology of [11], the relation $\trianglelefteq$ is *compatible* with h (in the case where $\trianglelefteq = \trianglelefteq'$).

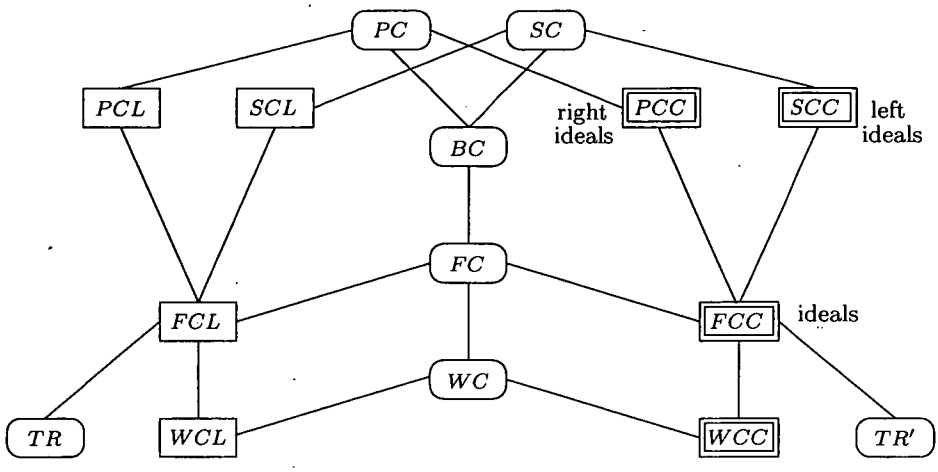


Figure 2: Classes of converse closed languages

Remark 9. The class FCC is not closed under either quotient. Let $\trianglelefteq$ be $\sqsubseteq$, let $\Sigma = \{a, b\}$, and let $L = \Sigma^* aba\Sigma^*$. Then L is $\sqsupseteq$-closed, but $K = a^{-1}L = \Sigma^* aba\Sigma^* + ba\Sigma^*$ is not, because $ba \in K$, but $bba \notin K$. Symmetrically, La^{-1} is not $\sqsupseteq$-closed.

Remark 10. No class XCC is closed under homomorphism. For let $\Sigma = \Delta = \{a, b\}$, $h(a) = h(b) = b$, and $L = \{\varepsilon, a\}$. Then $L \in XCL$ and $\overline{L} = (b + aa + ab)\Sigma^* = \Sigma^*(b + aa + ab) = \Sigma^*(b + aa + ab)\Sigma^* = \Sigma^* b\Sigma^* + \Sigma^* a\Sigma^* a\Sigma^* + \Sigma^* a\Sigma^* b\Sigma^* \in XCC$, for all $X \in \{P, S, F, W\}$. However, $h(\overline{L}) = bb^*$, and $K = \overline{h(\overline{L})} = \varepsilon + \Sigma^* a\Sigma^*$ is not in XCL, since $b \notin K$.

Remark 11. All the classes of the form XCC are closed under concatenation, because we have $(L\Sigma^*)(K\Sigma^*) = (L\Sigma^* K)\Sigma^*$, *etc.* Also, all the classes are closed under positive closure, which is defined as $L^+ = LL^*$, because $L\Sigma^* \supseteq L\Sigma^* L\Sigma^*$, *etc.* However, converse closed classes are not closed under star if $\epsilon \notin L$, because $\{\varepsilon\} \cup L\Sigma^*$ is not a right ideal, *etc.*

5.1 Transitive sofic languages

Factorial languages contain an interesting subclass which we discuss next; for more details we refer the reader to the literature [3, 4, 16]. A language $M \subseteq \Sigma^*$ is a *monoid* if it contains ε and is closed under concatenation. A monoid L is *very pure* if $uv, vu \in L$ implies $u, v \in L$. A factorial language is called *sofic* if it is regular. A language L is *transitive* if for all $u, w \in L$, there exists $x \in \Sigma^*$ such that $v = uxw \in L$. Let $F(L)$ be the set of all factors of words in L.

Transitive sofic languages constitute the class TR in Figure 2, and TR' is the class of their complements. The following characterization is given in [3]:

Proposition 14 (Béal & Perrin). *A language L is sofic and transitive if and only if there exists a very pure regular language M, which is a monoid, such that $L = F(M)$.*

Example 3. Let $\Sigma^* = \{a, b, c\}$, let $M = (ab^*c + b)^*$, and let $L = F(M)$. To find $F(M)$ we construct a nondeterministic finite automaton $\mathcal{N}$ from the minimal deterministic finite automaton $\mathcal{D}$ for M as follows. All the states of M, except the rejecting "dead" state, are made both accepting and initial. This guarantees that $\mathcal{N}$ accepts precisely all the factors of words of M. We then determinize $\mathcal{N}$ using the standard subset construction to obtain the minimal deterministic finite automaton $\mathcal{A}$ for $L = F(M)$. We leave the details to the reader; the automaton $\mathcal{D}$ has only three states, and $\mathcal{A}$ has only four. From $\mathcal{A}$ we can find the following regular expressions for L:

$$L = b^* + b^*c(b + ab^*c)^*(\varepsilon + ab^*) + b^*a(b + cb^*a)^*(\varepsilon + cb^*) = \overline{\Sigma^*(ab^*a + cb^*c)\Sigma^*}.$$

Here the language $G = ab^*c + b$ is a circular code [4] and is a minimal generating set of M. The monoid $M = G^*$ is very pure, and $L = F(M)$ is transitive.

Proposition 15. *Let $h : \Sigma^* \to \Delta^*$ be a homomorphism, let $K \subseteq h(\Sigma^*)$ and let $L = h^{-1}(K)$. If K is a transitive sofic language then so is L.*

Proof. Since K is regular, so is L, since regular languages are closed under inverse homomorphism. Suppose that u and w are in L, and let $h(u) = x$, $h(w) = z$. Since K is transitive, for every $x, z \in K$ there exists $y \in \Delta^*$ such that xyz is in K. Since K is factorial, we also have $y \in K$. Hence there exists $v \in L$ such that $h(v) = y$. Since $h(uvw) = h(u)h(v)h(w) = xyz \in K$, we also have $uvw \in L$, and we have shown that L is transitive. Finally, if $uvw \in L$ and $v \notin L$, then $h(uvw) \in K$ and $h(v) \notin K$, contradicting that K is factorial. Hence L is also factorial. Altogether, L is transitive sofic. □

Remark 12. Transitive sofic languages are not closed under left and right quotients, intersection, union, complement and concatenation. Let $\Sigma = \{a, b, c, d, e\}$, let L be the transitive sofic language L of Example 3, and let K be a similar language,

$$K = e^* + e^*c(e + de^*c)^*(\varepsilon + de^*) + b^*d(e + ce^*d)^*(\varepsilon + ce^*) = \overline{\Sigma^*(de^*d + ce^*c)\Sigma^*}.$$

Then $L \cap K = \varepsilon + c$, which is not transitive, because, for instance, $cxc \notin L \cap K$ for any $x \in \Sigma^*$. Also, for the language L of Example 3, $cac \in a^{-1}L$, but $a \notin a^{-1}L$; hence $a^{-1}L$ is not factorial. Similarly, $cac \in La^{-1}$, but $a \notin La^{-1}$; hence La^{-1} is not factorial. Moreover, let $\Sigma = \{a, b\}$, $K = a^*$, and $L = b^*$. Then K and L are transitive, but $K \cup L$ and KL are not. We have $a, b \in K \cup L$, but there is no $x \in \Sigma^*$ such that $axb \in K \cup L$. Also, $ab \in KL$, but there is no $x \in \Sigma^*$ such that $abxab \in L$. The complement of L is not factorial, since $\varepsilon \notin L$.

5.2 Containments involving converse closed languages

Proposition 16. *All containments shown in Figures 1 and 2 are proper, and there are no other containments, except those implied by transitivity.*

Proof. We consider the classes starting with the largest ones.

1. $PC \not\supseteq SCC$: $L_{11} = (a+b)^*(aa+aaba) \in SCC$, because L_{11} is a left ideal, but L_{11} is not prefix-convex because $aa, aaba \in L_{11}$, but $aab \notin L_{11}$.
2. $SC \not\supseteq PCC$: $L_{12} = (aa+abaa)(a+b)^* \in PCC \setminus SC$, by a similar argument.
3. $PCL \not\supseteq TR'$: $L_{13} = (a+b+c)^*(ab^*a+cb^*c)(a+b+c)^* \in TR' \setminus PCL$, because $aa \in L_{13}$, but $a \notin L_{13}$.
4. $PCL \not\supseteq WCC$: $L_{14} = (a+b)^*a(a+b)^*a(a+b)^* \in WCC \setminus PCL$, because $aa \in L_{14}$, but $a \notin L_{14}$.
5. $SCL \not\supseteq TR'$: $L_{13} \in TR' \setminus SCL$.
6. $SCL \not\supseteq WCC$: $L_{14} \in WCC \setminus SCL$.
7. $PF \not\supseteq TR'$: $L_{13} \in TR' \setminus PF$, because $aaa \in L_{13}$ and $aa \in L_{13}$.
8. $PF \not\supseteq WCC$: $L_{14} \in WCC \setminus PF$, because $aaa \in L_{14}$ and $aa \in L_{14}$.
9. $SF \not\supseteq TR'$: $L_{13} \in TR' \setminus SF$.
10. $SF \not\supseteq WCC$: $L_{14} \in WCC \setminus SF$.
11. $PCC \not\supseteq WCL$: $L_4 = \{\varepsilon, a\} \in WCL \setminus PCC$.
12. $PCC \not\supseteq TR$: $L_{15} \in TR \setminus PCC$, where L_{15} is L from Example 3, because $a \in L_{15}$, but $aa \notin L_{15}$.
13. $PCC \not\supseteq WF$: $L_8 = \{a\} \in WF \setminus PCC$.
14. $SCC \not\supseteq WCL$: $L_4 = \{\varepsilon, a\} \in WCL \setminus SCC$.
15. $SCC \not\supseteq TR$: $L_{15} \in TR \setminus SCC$.
16. $SCC \not\supseteq WF$: $L_8 = \{a\} \in WF \setminus SCC$.
17. $FCC \not\supseteq PCC$: $L_{16} = a(a+b)^* \in PCC \setminus FCC$.
18. $FCC \not\supseteq SCC$: $L_{17} = (a+b)^*a \in SCC \setminus FCC$.
19. $WCC \not\supseteq FCC$: $L_{18} = (a+b)^*aa(a+b)^* \in FCC \setminus WCC$, since $aa \in L_{18}$, but $aba \notin L_{18}$.
20. $TR' \not\supseteq WCC$: $L_{14} \in WCC \setminus TR'$, because $\overline{L_{14}} = b^* + b^*ab^*$ is not transitive, since it has no word axa for any $x \in \Sigma^*$.
21. $WCC \not\supseteq TR'$: $L_{13} \in TR' \setminus WCC$, because $aa \in L_{13}$, but $aca \notin L_{13}$.

Hence all the classes shown in the two figures are distinct and there are no other containments. □

6 Concatenation in free and closed languages

The next example illustrates that, in general, $\trianglelefteq$-closed and $\trianglelefteq$-free languages are not closed under concatenation.

Example 4. Suppose $u \trianglelefteq v$ if and only if either $u = v$ or $|u| = |v|$ and u precedes v in the lexicographic order. Thus, for $\Sigma = \{a, b\}$, we have $a \lhd b$, $aa \lhd ab \lhd ba \lhd bb$, $aaa \lhd aab \lhd aba \lhd \cdots \lhd bbb$, *etc.* Let $K = \{a, bb\}$; then K is $\trianglelefteq$-free. However, $KK = \{aa, abb, bba, bbbb\}$ is not. Also, if $L = \{aa, ab\}$, then L is $\trianglelefteq$-closed. However, $LL = \{aaaa, aaab, abaa, abab\}$ is not. Hence, for this binary relation, the classes of $\trianglelefteq$-closed and $\trianglelefteq$-free languages are not closed under concatenation.

6.1 Free languages and concatenation

A binary relation $\trianglelefteq$ is *propagating* if $x_1x_2 \lhd y_1y_2$ implies that

$$(x_1 \lhd y_1) \vee (y_1 \lhd x_1) \vee (x_2 \lhd y_2) \vee (y_2 \lhd x_2),$$

for all $x_1, x_2, y_1, y_2 \in \Sigma^*$, where $\vee$ denotes disjunction.

Proposition 17. *If $\trianglelefteq$ is propagating, and K and L are $\trianglelefteq$-free, then so is KL.*

Proof. Suppose K and L are $\trianglelefteq$-free, but $M = KL$ is not. Then there are $x_1, y_1 \in K$, $x_2, y_2 \in L$ such that $x_1x_2 \lhd y_1y_2$. Since $\trianglelefteq$ is propagating, either x_1 and y_1 are unequal and comparable under $\trianglelefteq$, or x_2 and y_2 are. Thus either K or L is not $\trianglelefteq$-free, which is a contradiction. □

Lemma 1. *The binary relations $\leq$, $\preceq$, $\sqsubseteq$ and $\Subset$ are propagating.*

Proof. Suppose $x_1x_2 < y_1y_2$; then $x_1x_2v = y_1y_2$, where $v \in \Sigma^*$ is nonempty. If $x_1 < y_1$ or $x_1 > y_1$, the condition of the lemma is satisfied. If $x_1 = y_1$, then $x_2 < y_2$, and the lemma holds. A symmetric argument works for $\preceq$.

Suppose $x_1x_2 \sqsubset y_1y_2$; then $ux_1x_2v = y_1y_2$, for some $u, v \in \Sigma^*$, where $uv \neq \varepsilon$. If $ux_1 < y_1$, then $x_1 \sqsubset y_1$. If $ux_1 > y_1$, then $x_2 \sqsubset y_2$. If $ux_1 = y_1$ and $u \neq \varepsilon$, then $x_1 \sqsubset y_1$. If $ux_1 = y_1$ and $u = \varepsilon$, then $x_1 = y_1$, and $x_2 \sqsubset y_2$, since $v \neq \varepsilon$.

Now suppose that $x_1x_2 \Subset y_1y_2$; then $x_1 = a_1 \cdots a_j$, $x_2 = a_{j+1} \cdots a_n$, for some j, and $y_1 = v_0a_1v_1 \cdots a_iv_i'$ and $y_2 = v_i''a_{i+1}v_{i+1} \cdots a_nv_n$, for some i, where $v_i = v_i'v_i''$, $v_0, \ldots, v_n \in \Sigma^*$, $a_1, \ldots, a_n \in \Sigma$, and $v_1 \cdots v_n \neq \varepsilon$. If $j < i$, then $x_1 \Subset y_1$. If $j > i$, then $x_2 \Subset y_2$. If $j = i$, and $v_0v_1 \cdots v_i' \neq \varepsilon$, then $x_1 \Subset y_1$. If $j = i$, and $v_0v_1 \cdots v_i' = \varepsilon$, then $x_2 \Subset y_2$. □

Corollary 8. *The prefix-, suffix-, bifix-, factor-, and subword-free classes are closed under concatenation.*

6.2 Closed languages and concatenation and star

We now consider $\trianglelefteq$-closed languages. A binary relation $\trianglelefteq$ is *factoring* if $x \trianglelefteq y_1y_2$ implies that $x = x_1x_2$ for some $x_1, x_2 \in \Sigma^*$ such that $x_1 \trianglelefteq y_1$, $x_2 \trianglelefteq y_2$.

Proposition 18. *If $\trianglelefteq$ is factoring, and K and L are $\trianglelefteq$-closed, then so is KL.*

Proof. Suppose K and L are $\trianglelefteq$-closed, but $M = KL$ is not. Then there exist $x \notin M$, $y_1 \in K$, $y_2 \in L$ such that $x \triangleleft y_1y_2$. Since $\trianglelefteq$ is factoring, $x = x_1x_2$, where $x_1 \trianglelefteq y_1$ and $x_2 \trianglelefteq y_2$. If K and L are $\trianglelefteq$-closed, then $x_1 \in K$, $x_2 \in L$, and $x \in M$—a contradiction. □

Lemma 2. *The binary relations $\leq$, $\preceq$, $\sqsubseteq$ and $\Subset$ are factoring.*

Proof. Suppose $x \leq y_1y_2$; then $xv = y_1y_2$ for some $v \in \Sigma^*$. For $x \leq y_1$, since $\varepsilon \leq y_2$, we have $x_1 = x$, and $x_2 = \varepsilon$. If $x > y_1$, then $x = x_1x_2$, where $x_1 = y_1$ and $x_2v = y_2$. Then $x_1 \leq y_1$, and $x_2 \leq y_2$. A symmetric argument works for $\preceq$.

Suppose $x \sqsubseteq y_1y_2$; then $uxv = y_1y_2$, for some $u, v \in \Sigma^*$. If $ux \leq y_1$, then $x_1 = x \sqsubseteq y_1$ and $x_2 = \varepsilon \sqsubseteq y_2$. If $ux > y_1$ and $u < y_1$, then $x = x_1x_2$, where $ux_1 = y_1$ and $x_2v = y_2$. Then $x_1 \sqsubseteq y_1$, and $x_2 \sqsubseteq y_2$. If $ux > y_1$ and $u \geq y_1$, then $x_1 = \varepsilon \sqsubseteq y_1$ and $x_2 = x \sqsubseteq y_2$.

Now suppose that $x \Subset y_1y_2 = v$; then $x = a_1 \cdots a_n$ and $v = v_0a_1v_1 \cdots a_nv_n$, where $v_0, \ldots, v_n \in \Sigma^*$, $a_1, \ldots, a_n \in \Sigma$, and, for some i we have $y_1 = v_0a_1v_1 \cdots a_iv_i'$ and $y_2 = v_i''a_{i+1}v_{i+1} \cdots a_nv_n$, where $v_i = v_i'v_i''$. If $i = n$, then $x_1 = x \Subset y_1$ and $x_2 = \varepsilon \Subset y_2$. If $i < n$, then $x = x_1x_2$, where $x_1 = a_1 \cdots a_i \Subset y_1$ and $x_2 = a_{i+1} \cdots a_n \Subset y_2$. □

Corollary 9. *The prefix-, suffix-, bifix- (= factor-), and subword-closed classes are closed under concatenation.*

A binary relation $\trianglelefteq$ is *ε-full* if $\varepsilon \trianglelefteq w$ for all $w \in \Sigma^*$. Note that all our example relations are ε-full.

Proposition 19. *If $\trianglelefteq$ is antisymmetric, factoring, and ε-full, and L is $\trianglelefteq$-closed, then so is L^*.*

Proof. We adapt Thierrin's proof [20] given for the case where $\trianglelefteq$ is the subword relation. Suppose L is $\trianglelefteq$-closed. If $L = \emptyset$, then $L^* = \{\varepsilon\}$. If there is no $w \in \Sigma^*$, $w \neq \varepsilon$, such that $w \trianglelefteq \varepsilon$, then L^* is $\trianglelefteq$-closed. If there is such a w, then $\varepsilon \trianglelefteq w$, since $\trianglelefteq$ is ε-full. However, this contradicts the antisymmetry of $\trianglelefteq$. Thus, if $L = \emptyset$, then L^* is $\trianglelefteq$-closed. Therefore assume that $L \neq \emptyset$. Since $\trianglelefteq$ is ε-full, we must have $\varepsilon \in L$. We argue that L^* is $\trianglelefteq$-closed if L^n is $\trianglelefteq$-closed for each $n \geq 0$. As we have shown above, $L^0 = \{\varepsilon\}$ is $\trianglelefteq$-closed. If $n = 1$, then $L^n = L$, and L is $\trianglelefteq$-closed by assumption. For $n > 1$, since $\trianglelefteq$ is factoring, L^n is $\trianglelefteq$-closed by Proposition 18. Since the union of $\trianglelefteq$-closed languages is $\trianglelefteq$-closed by Proposition 11, we have our result. □

Corollary 10. *The prefix-, suffix-, bifix- (= factor-), and subword-closed classes are closed under star.*

7 Conclusions

We have provided a common framework for several classes of languages, and we have shown that closure properties of these classes can be studied using binary relations on Σ^*. Table 2 summarizes our closure results. In case closure holds only for some of the relations, we specify these relations in the table.

Table 2: Closure results for prefix, suffix, factor, and subword relations.

	convex	*closed*	*converse closed*	*free*
intersection	yes	yes	yes	yes
union	no	yes	yes	no
complement	no	no	no	no
concatenation	no	yes	yes	yes
Kleene star	no	yes	no	no
positive closure	no	yes	yes	no
left quotient	prefix	prefix	prefix	prefix
	subword	subword	subword	subword
right quotient	suffix	suffix	suffix	suffix
	subword	subword	subword	subword
homomorphism	no	no	no	no
inverse homomorphism	yes	yes	yes	yes

The problems of deciding whether a language specified by a deterministic or nondeterministic finite automaton is prefix-, suffix-, factor-, or subword-convex, -free, or -closed have been recently studied in [8].

Acknowledgment: We thank Larry Cummings, Helmut Jürgensen, Jacques Sakarovitch and Jeff Shallit for useful comments and pointers to references. We are grateful to two anonymous referees for suggesting numerous improvements.

References

[1] Ang, T. and Brzozowski, J. A. Continuous Languages. In Csuhaj-Varjú, E. and Ésik, Z., editors, *Proc. 12th Int. Conference on Automata and Formal Languages,* Computer and Automation Research Institute, Hungarian Academy of Sciences, 74–85, 2008.

[2] Avgustinovich, S. V. and Frid, A. E. A unique decomposition theorem for factorial languages. *Internat. J. Algebra Comput.,* (15): 149–160, 2005.

[3] Béal, M. P. and Perrin, D. Une caractérisation des ensembles sofiques. *C. R. Acad. Sci., Paris,* (303): 255–257, 1986.

[4] Berstel, J. and Perrin, D. *Theory of Codes.* Academic Press, 1985.

[5] Brzozowski, J. A. Representation of a class of nondeterministic semiautomata by canonical words. *Theoret. Comput. Sci.,* (356): 46–57, 2006.

[6] Brzozowski, J. A. and Jürgensen, H. Representation of semiautomata by canonical words and equivalences. *Internat. J. Found. Comput. Sci.,* (16): 831–850, 2005.

[7] Brzozowski, J. A. and Jürgensen, H. Representation of semiautomata by canonical words and equivalences, Part II: Applications to the specification of software modules. *Internat. J. Found. Comput. Sci.,* (18): 1065–1087, 2007.

[8] Brzozowski, J. A., Shallit, J. O. and Xu, Z. Decision problems for convex languages. In Dediu, A. H., Ionescu, A. M. and Martin-Vide, C., editors, *Proc. 3rd Int. Conference on Languages and Automata Theory and Applications,* LNCS (5457): 247–258. Springer, 2009.

[9] Guo, Y. Q., Shyr, H. J. and Thierrin, G. E-convex infix codes. *Order,* (3): 55–59,1986.

[10] Haines, L. H. On free monoids partially ordered by embedding. *J. Combin. Theory,* (6): 94–98,1969.

[11] Jürgensen, H., Kari, L. and Thierrin, G. Morphisms preserving densities. *Internat. J. Comput. Math.,* (78): 165–189, 2001.

[12] Jürgensen, H. and Konstantinidis, S. Codes. In Rozenberg, G. and Salomaa, A., editors: *Handbook of Formal Languages,* (1): 511–607. Springer, 1997.

[13] Jürgensen, H. and Yu, S. S. Relations on free monoids, their independent sets, and codes. *Internat. J. Comput. Math.,* (40): 17–46, 1991.

[14] Kruskal, J. B. The theory of well-quasi-ordering: a frequently discovered concept. *J. Combin. Theory,* (A) (13): 297–305,1972.

[15] De Luca, A. and Varricchio, S. Some combinatorial properties of factorial languages. In Capocelli, R., editor, *Sequences,* 258–266. Springer, 1990.

[16] Restivo, A. Finitely generated sofic systems. *Theoret. Comput. Sci.,* (65): 265–270, 1989.

[17] Shur, A. M. Factorial languages of low combinatorial complexity. In Ibarra, O. H. and Dang, Z., editors, *Proc. Developments in Language Theory,* LNCS (4036): 397–407. Springer, 2006.

[18] Shyr, H. J. *Free Monoids and Languages.* Hon Min Book Co., Taichung, Taiwan, 2001.

[19] Shyr, H. J. and Thierrin, G. Hypercodes. *Inform. and Control,* (24): 45–54, 1974.

[20] Thierrin, G. Convex languages. In Nivat, M., editor, *Automatâ, Languages and Programming,* 481–492. North-Holland, 1973.

Received 25th September 2008

Acta Cybernetica **19** (2009) 465–477.

Counting Distinct Squares in Partial Words*

F. Blanchet-Sadri† Robert Mercaş‡ and Geoffrey Scott§

Abstract

A well known result of Fraenkel and Simpson states that the number of distinct squares in a word of length n is bounded by $2n$ since at each position there are at most two distinct squares whose last occurrence start. In this paper, we investigate the problem of counting distinct squares in partial words, or sequences over a finite alphabet that may have some "do not know" symbols or "holes" (a (full) word is just a partial word without holes). A square in a partial word over a given alphabet has the form uu' where u' is *compatible* with u, and consequently, such square is compatible with a number of full words over the alphabet that are squares. We consider the number of distinct full squares compatible with factors in a partial word with h holes of length n over a k-letter alphabet, and show that this number increases polynomially with respect to k in contrast with full words, and give bounds in a number of cases. For partial words with one hole, it turns out that there may be more than two squares that have their last occurrence starting at the same position. We prove that if such is the case, then the hole is in the shortest square. We also construct a partial word with one hole over a k-letter alphabet that has more than k squares whose last occurrence start at position zero.

Keywords: combinatorics on words, partial words, squares

1 Introduction

Computing repetitions such as squares in sequences or strings of symbols from a finite alphabet is profoundly connected to numerous fields such as biology, computer

*This material is based upon work supported by the National Science Foundation under Grant No. DMS–0452020. This work was done during the second author's stay at the University of North Carolina at Greensboro. A World Wide Web site has been created at `www.uncg.edu/cmp/research/freeness` for this research.

†Department of Computer Science, University of North Carolina, P.O. Box 26170, Greensboro, NC 27402–6170, USA, E-mail: `blanchet@uncg.edu`

‡GRLMC, Universitat Rovira i Virgili, Plaça Imperial Tárraco, 1, Tarragona, 43005, Spain and MOCALC Research Group, Faculty of Mathematics and Computer Science, University of Bucharest, Academiei, 14, 010014, Bucharest, Romania

§Department of Mathematics, Dartmouth College, 6188 Kemeny Hall, Hanover, NH 03755–3551, USA

science, and mathematics [8]. The stimulus for recent works on repetitions in strings is the study of biological sequences such as DNA that play a central role in molecular biology. In addition to its sheer quantity, repetitive DNA is striking for the variety of repetitions it contains, for the various proposed mechanisms explaining the origin and maintenance of repetitions, and for the biological functions that some of the repetitions may play. The literature has generally considered problems in which a period u of a repetition is invariant. It has been required that occurrences of u match each other exactly. In some applications however, such as DNA sequence analysis, it becomes interesting to relax this condition and to recognize u' as an occurrence of u if u' is *compatible* with u.

A well known result of Fraenkel and Simpson [3] states that the number of distinct squares in a word of length n is bounded by $2n$ since at each position there are at most two distinct squares whose last occurrence start. In [6], Ilie improves this bound to $2n - \Theta(\log n)$. Based on numerical evidence, it has been conjectured that this number is actually less than n. In this paper, we investigate the problem of counting distinct squares in partial words, or sequences over a finite alphabet that may contain some "do not know" symbols or "holes." In Section 2, after making some remarks about the maximum number of distinct full squares compatible with factors of a partial word, we give some lower bounds for that number. These bounds are related to the length of the word, the alphabet size this word is defined on, and the number of holes it contains. In Section 3, we show that for partial words with one hole, there may be more than two squares that have their last occurrence starting at the same position. We prove that if such is the case, then the hole is in the shortest square. There, we also construct for $k \geq 2$, a partial word with one hole over a k-letter alphabet that has more than k squares whose last occurrence start at position 0. Finally in Section 4, we provide some conclusions and suggestions for future work.

We end this section by reviewing basic concepts on partial words. Fixing a nonempty finite set of letters or an *alphabet* A, a *partial word* u of length $|u| = n$ over A is a partial function $u : \{0, \ldots, n-1\} \to A$. For $0 \leq i < n$, if $u(i)$ is defined, then i belongs to the *domain* of u, denoted by $i \in D(u)$, otherwise i belongs to the *set of holes* of u, denoted by $i \in H(u)$. The unique word of length 0, denoted by ε, is called the *empty* word. For convenience, we will refer to a partial word over A as a word over the enlarged alphabet $A_\diamond = A \cup \{\diamond\}$, where $\diamond \notin A$ represents a hole. The set of all words (respectively, partial words) over A of finite length is denoted by A^* (respectively, $A_\diamond^*$).

The partial word u is *contained in* the partial word v, denoted by $u \subset v$, provided that $|u| = |v|$, all elements in $D(u)$ are in $D(v)$, and for all $i \in D(u)$ we have that $u(i) = v(i)$. As a weaker notion, u and v are *compatible*, denoted by $u \uparrow v$, provided that there exists a partial word w such that $u \subset w$ and $v \subset w$. An equivalent formulation of compatibility is that $|u| = |v|$ and for all $i \in D(u) \cap D(v)$ we have that $u(i) = v(i)$. We denote by $u \vee v$ the least upper bound of u and v, that is, for every partial word w such that $u \subset w$ and $v \subset w$, we have $(u \vee v) \subset w$. If $u \not\uparrow v$, then we adopt the convention that $u \vee v = \varepsilon$. The following rules are useful for computing with partial words: (1) *Multiplication:* If $u \uparrow v$ and $x \uparrow y$,

then $ux \uparrow vy$; (2) *Simplification:* If $ux \uparrow vy$ and $|u| = |v|$, then $u \uparrow v$ and $x \uparrow y$; and (3) *Weakening:* If $u \uparrow v$ and $w \subset u$, then $w \uparrow v$.

A partial word u is *primitive* if there exists no word v such that $u \subset v^n$ with $n \geq 2$. If u is a nonempty partial word, then there exist a primitive word v and a positive integer n such that $u \subset v^n$. Uniqueness holds for full words but not for partial words as seen with $u = \diamond a$ where $u \subset a^2$ and $u \subset ba$ for distinct letters a, b. For partial words u, v, w, if $w = uv$, then u is a *prefix* of w, denoted by $u \leq w$, and if $v \neq \varepsilon$, then u is a *proper prefix* of w, denoted by $u < w$. If $w = xuy$, then u is a *factor* of w. If $u = u_1u_2$ for some nonempty compatible partial words u_1 and u_2, then u is called a *square*. Whenever we refer to a square u_1u_2 it will imply that $u_1 \uparrow u_2$.

2 Counting distinct squares: A first approach

In a full word, every factor of length $2n$ contains at most one square factor ww with $|w| = n$. In a square partial word w_0w_1 where $w_0 \uparrow w_1$, we call the word $v = w_0 \vee w_1$ the *general form* of the square. For example, the general form of the square $ab\diamond\diamond c\diamond a\diamond d\diamond\diamond\diamond$ is $abd\diamond c\diamond$. We observe that in partial words, a square w_0w_1 may be compatible with more than one distinct full square of length $2|w_0|$. For example, the word $aa\diamond aa\diamond$ over the alphabet $\{a, b, c\}$ is compatible with three distinct full squares of length 6: $(aaa)^2$, $(aab)^2$ and $(aac)^2$. It is easy to see that if $aa\diamond aa\diamond$ is a word over an alphabet of size k, then it is compatible with exactly k squares of length 6. Whenever we talk about a full square compatible with a general form, we refer to a square that has the first half compatible with the general form. In general, if $w = a_0a_1 \ldots a_{2m-1}$ is a partial word over a k-letter alphabet A, and w is a square, then w is compatible with exactly $k^{\|H(v)\|}$ squared full words of length m, where $v = a_0a_1 \ldots a_{m-1} \vee a_ma_{m+1} \ldots a_{2m-1}$.

At this point, we see that the study of distinct squares in partial words is quite different from the study of distinct squares in full words. In the case of full words, there exists an upper bound for the number of distinct squares in a word of length n, no matter what the alphabet size is. The same statement is certainly untrue for partial words. For example, the number of distinct nonempty full squares compatible with $\diamond\diamond$ is equal to k, where k is the alphabet size.

Let w be a partial word over a k-letter alphabet A. We will denote by $f_k(w)$ the number of distinct nonempty full squares over A compatible with factors of w, and by $g_{h,k}(n)$ the maximum of the $f_k(w)$'s where w ranges over all partial words of length n with h holes, over alphabet A. Note that the number of all distinct full square nonempty words compatible with factors of $\diamond^n$, where n is a positive integer, over A, is equal to the number of all distinct full nonempty words of length $i \leq \lfloor \frac{n}{2} \rfloor$ over A. Using this remark,

$$g_{n,k}(n) = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} k^i = \frac{k\,(k^{\lfloor \frac{n}{2} \rfloor} - 1)}{k-1} \tag{1}$$

Note that if n is odd, then $g_{n-1,k}(n-1) = g_{n,k}(n)$ and $g_{n-1,k}(n) = g_{n,k}(n)$. The first equality follows directly from (1). For the second equality, note that the number of distinct nonempty full squares compatible with factors of $\diamond^{n-1}a$ over the k-letter alphabet A where $a \in A$ is at least $g_{n-1,k}(n-1) = g_{n,k}(n)$ (those compatible with factors of $\diamond^{n-1}$). Thus, $g_{n-1,k}(n) \geq g_{n,k}(n)$. Since the function $g_{h,k}(n)$ is clearly monotonically increasing with respect to h, k, and n, it follows that $g_{n-1,k}(n) \leq g_{n,k}(n)$. Thus, $g_{n-1,k}(n) = g_{n,k}(n)$.

As we have seen earlier with the word $\diamond\diamond$, the number of distinct nonempty full squares compatible with factors of a partial word may be unbounded if we allow the alphabet size to grow arbitrarily large. However, we can often write this number as a function of the alphabet size. The following proposition shows that this number is indeed a polynomial in the alphabet size.

Proposition 1. *Let w be a partial word of length n over a k-letter alphabet, and let S_1 be the set of general forms of all factors of w that are squares. Let S_m be the set of all partial words v that can be written as $v = u_0 \vee u_1 \vee \cdots \vee u_{m-1}$, where $u_i \in S_1$ for all $0 \leq i < m$ and $u_i \neq u_j$ for all $i < j < m$. Then the number of full distinct squares compatible with factors of w is given by*

$$\sum_{m=1}^{\lfloor \frac{n}{2} \rfloor} ((-1)^{m-1} \sum_{s \in S_m} k^{\|H(s)\|}) \tag{2}$$

Proof. For a set X of partial words, denote by $\hat{X}$ the set of all full words compatible with elements of X. The number of full distinct square words compatible with factors of w is given by $\|\hat{S_1}\|$. By the principle of inclusion-exclusion,

$$\hat{S_1} = \sum_{m=1}^{\lfloor \frac{n}{2} \rfloor} ((-1)^{m-1} \sum_{s \in S_m} \|\{\hat{s}\}\|)$$

Since $\|\{\hat{s}\}\| = k^{\|H(s)\|}$, the proof is complete. □

To generalize the study of counting distinct squares in words to partial words, we are interested in the limit behaviour of $g_{h,k}(n)$ as k increases. However, as we have seen with the word $w = \diamond\diamond$, the value $\lim_{k\to\infty} f_k(w)$ may be infinity. Following Proposition 1, if we treat k as an unknown variable, the number of distinct nonempty full squares compatible with factors in any partial word is a polynomial with respect to k. If we consider all such polynomials corresponding to words of length n containing h holes, the maximal such polynomial would describe this limiting behavior. Given a finite length n, there exist only finitely many partial words of length n up to an isomorphism between letters. Therefore, a lower bound for $g_{h,k}(n)$ can be given using the leading term of this well defined maximal polynomial, $m_{h,k}(n)$.

The next results give bounds on the leading term in $m_{h,k}(n)$. We begin by defining a *free hole* of a square. Let w be a partial word over an alphabet A that

contains a factor v that is a square. A hole in v is called a *free hole* of v if the square v is preserved even after we replace the hole with any letter of A. For example, consider the partial word $w = ab\diamond a\diamond\underline{\diamond}$ over the alphabet $\{a, b, c\}$. The underlined hole is a free hole of the squares $ab\diamond a\diamond\diamond$ and $\diamond\diamond$, but not of $\diamond a\diamond\diamond$. It is easy to see that the number of free holes of a square factor is exactly twice the number of holes in the general form of that square. Two free holes in positions i and j in a square v are aligned if $i = j + \frac{|v|}{2}$ or $j = i + \frac{|v|}{2}$ and $v(i) = v(j) = \diamond$.

Note that the degree of $m_{h,k}(n)$ is $\lfloor \frac{h}{2} \rfloor$. To see this, let w be a word of length n with h holes over a k-letter alphabet. Clearly, any factor of w that is a square has at most $\lfloor \frac{h}{2} \rfloor$ holes in its general form. Thus, by (2) there can be no term of $m_{h,k}(n)$ with k raised to a power higher than $\lfloor \frac{h}{2} \rfloor$. Also note that the word $w = \diamond^h a^{n-h}$ achieves this bound. The following technical lemma will assist us in proving results about the coefficients of $m_{h,k}(n)$.

Lemma 1. *Let l be a positive integer, let w be a partial word of length n, and let $0 \leq p_1 \leq p_2 < n$. Then there are at most $\lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 1$ factors $v = w(i)w(i+1)\dots w(i+2l-1)$ of length $2l$ in w such that $i \leq p_1$ and $i + l > p_2$.*

Proof. Assume that there exist $\lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 2$ such factors of length $2l$ in w. Since all of these factors have the same length, no two of them may start at the same position. Therefore, $p_1 \geq \lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 1$. In particular, one of these factors must start at a position no later than $p_1 - (\lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 1)$. This gives us that $l > ((p_2 - p_1) + \lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 1)$ from the condition that $i + l > p_2$. For any factor $v = w(i)w(i+1)\dots w(i+2l-1)$ of length $2l$ in w, we know that the length of w must exceed $2l + i$. Since there exist $\lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 2$ such factors, at least one must start at a position i satisfying $i \geq \lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 1$. Therefore, we obtain the contradiction

$$n \geq 2(p_2 - p_1 + \lfloor \frac{n - 2(p_2 - p_1 + 1)}{3} \rfloor + 2) + \lfloor \frac{n - 2(p_2 - p_1 + 1)}{3} \rfloor + 1$$

$$n \geq 3\lfloor \frac{n - 2(p_2 - p_1 + 1)}{3} \rfloor + 2(p_2 - p_1 + 1) + 3$$

$$n \geq n - 2(p_2 - p_1 + 1) - 2 + 2(p_2 - p_1 + 1) + 3$$

□

Intuitively, the above lemma states that for any $l > 0$, there can be at most $\lfloor \frac{n-2(p_2-p_1+1)}{3} \rfloor + 1$ factors of length $2l$ that use the letters $w(p_1)w(p_1+1)\dots w(p_2)$ in their first half. We will use this lemma to find upper bounds for the leading term of $m_{h,k}(n)$.

Theorem 1. *The leading term in $m_{2h,k}(n)$ is $(\lfloor \frac{n-2h}{3} \rfloor + 1)k^h$.*

Proof. The degree of $m_{2h,k}(n)$ being h, it only remains to show that the coefficient of k^h in $m_{2h,k}(n)$ is equal to $\lfloor \frac{n-2h}{3} \rfloor + 1$. We will give a lower bound of this

coefficient by constructing a word with the given leading term. Consider any word w of length n containing $2h$ holes and the factor

$$a^{\lfloor \frac{n-2h}{3} \rfloor} \diamond^h a^{\lfloor \frac{n-2h}{3} \rfloor} \diamond^h a^{\lfloor \frac{n-2h}{3} \rfloor}$$

The following is an exhaustive list of general forms of factors of w that are squares containing $2h$ free holes:

$$\begin{array}{ccccc} aaa & \dots & aa\diamond\diamond & \dots & \diamond\diamond \\ aaa & \dots & a\diamond\diamond\diamond & \dots & \diamond a \\ & & \vdots & & \\ a\diamond\diamond & \dots & \diamond\diamond aa & \dots & aa \\ \diamond\diamond\diamond & \dots & \diamond aaa & \dots & aa \end{array}$$

These $\lfloor \frac{n-2h}{3} \rfloor + 1$ partial words are pairwise compatible, but for any words v_1, v_2 in the above list, $\|H(v_1 \vee v_2)\| < h$. Therefore, by (2) we see that the coefficient of k^h in $m_{2h,k}(n)$ will be at least $\lfloor \frac{n-2h}{3} \rfloor + 1$.

Note that the coefficient of k^h corresponding to a word w is equal to the number of distinct factors in w, that are squares with $2h$ free holes. Let

$$w = w_0 \diamond_0 w_1 \diamond_1 w_2 \diamond_2 \dots \diamond_{2h-1} w_{2h}$$

where $w_i \in A^*$ for all $0 \leq i \leq 2h$ and $\diamond_i = \diamond$ for all $0 \leq i < 2h$. Note that all factors of w with $2h$ free holes that are squares must have the same length (because in a square the free hole $\diamond_0$ is aligned with $\diamond_h$, the length of all such square factors will be twice the distance between $\diamond_0$ and $\diamond_h$). We observe that all factors of w that are squares containing $2h$ free holes must contain the first h holes of w in their first half. Therefore, every such factor contains $\diamond_0 w_1 \diamond_1 \dots \diamond_{h-1}$ in its first half. The length of $\diamond_0 w_1 \diamond_1 \dots \diamond_{h-1}$ is at least h, so by Lemma 1, there exist at most $\lfloor \frac{n-2h}{3} \rfloor + 1$ such factors. □

Proposition 2. *The leading term in $m_{2h+1,k}(n)$ is at least $(2\lfloor \frac{n-2h}{3} \rfloor + 1)k^h$.*

Proof. The degree of $m_{2h+1,k}(n)$ being h, it only remains to show that the coefficient of k^h in $m_{2h+1,k}(n)$ is at least $2\lfloor \frac{n-2h}{3} \rfloor + 1$. Consider any word w of length n containing $2h+1$ holes and the factor

$$a^{\lfloor \frac{n-2h}{3} \rfloor} \diamond^h a^{\lfloor \frac{n-2h}{3} \rfloor - 1} \diamond^{h+1} a^{\lfloor \frac{n-2h}{3} \rfloor}$$

The following is an exhaustive list of general forms of factors of w that are squares containing $2h$ free holes:

$$\begin{array}{ll} a^{\lfloor \frac{n-2h}{3} \rfloor - 1} a \diamond^{h-1} \diamond & a^{\lfloor \frac{n-2h}{3} \rfloor - 2} a \diamond^{h-1} \diamond \\ a^{\lfloor \frac{n-2h}{3} \rfloor - 1} \diamond\diamond^{h-1} a & a^{\lfloor \frac{n-2h}{3} \rfloor - 2} \diamond\diamond^{h-1} a \\ \vdots & \vdots \\ a \diamond^{h-1} \diamond a^{\lfloor \frac{n-2h}{3} \rfloor - 1} & \diamond^{h-1} \diamond aa^{\lfloor \frac{n-2h}{3} \rfloor - 2} \\ \diamond^{h-1} \diamond a^{\lfloor \frac{n-2h}{3} \rfloor - 1} a & \end{array}$$

There are $\lfloor\frac{n-2h}{3}\rfloor + 1$ words in the left column and $\lfloor\frac{n-2h}{3}\rfloor$ words in the right column. It is easy to check that if we select two compatible words v_1, v_2 from the above list of $(2\lfloor\frac{n-2h}{3}\rfloor + 1)$ partial words, $\|H(v_1 \vee v_2)\| < h$. Using (2) we get that the coefficient of k^h in $m_{2h+1,k}(n)$ will be at least $2\lfloor\frac{n-2h}{3}\rfloor + 1$. □

Proposition 3. *The leading term in $m_{2h+1,k}(n)$ is at most $(2\lfloor\frac{n-2h}{3}\rfloor + 3)k^h$ for $h > 1$.*

Proof. Let w be a word of length n containing $2h+1$ holes for some $h > 1$. Then w is of the form $w_0\diamond_0 w_1\diamond_1 w_2\diamond_2 \ldots \diamond_{2h} w_{2h+1}$ where $\diamond_i = \diamond$ for all i. We need to count the number of distinct factors of w that are squares containing $2h$ free holes. Let S denote the set of all such factors in w. Note that for every $s \in S$, there exists a hole in w that is not a free hole of s. Let S_j denote the set of all $s \in S$ having the property that $\diamond_j$ is not a free hole of s. Clearly, we have the partition $S = \cup_{0\le j\le 2h} S_j$.

First, assume that there exists $j \notin \{0, h, 2h\}$ such that $S_j \neq \emptyset$. Then $w_j\diamond_j w_{j+1} \uparrow w_k$ for some $j \neq k$. If there exists an i distinct from j such that $S_i \neq \emptyset$, then in one of the squares of S_i, the hole $\diamond_j$ is aligned with $\diamond_{k-1}$ or $\diamond_k$. In these cases, we get that $|w_{j+1}| \geq |w_k|$ or $|w_j| \geq |w_k|$ respectively. Both cases contradict with $w_j\diamond_j w_{j+1} \uparrow w_k$. Thus, $S_i = \emptyset$ for all $i \neq j$. Hence, we can replace $w_j\diamond w_{j+1}$ in w with w_k and preserve all squares. The resulting word has only $2h$ holes. From Theorem 1,

$$\|S\| \leq \lfloor\frac{n-2h}{3}\rfloor + 1$$

Next, let us consider the case where $S_j = \emptyset$ for every $j \notin \{0, h, 2h\}$. Note that all squares in S_0 have length equal to the distance between $\diamond_1$ and $\diamond_{h+1}$ in w, since these two holes are aligned in each square of S_0. Using the same argument, all squares in S_{2h} have length equal to the distance between $\diamond_1$ and $\diamond_{h+1}$ in w. Therefore, the length of squares in S_0 is equal to the length of the squares in S_{2h}. Note that all squares in S_0 and S_{2h} contain the factor $\diamond_1 w_2\diamond_2 \ldots \diamond_{h-1}$ in their first half. The length of this common factor is at least $h-1$. By Lemma 1, $\|S_0 \cup S_{2h}\| \leq \lfloor\frac{n-2(h-1)}{3}\rfloor + 1 = \lfloor\frac{n-2h+5}{3}\rfloor$. Since all squares in S_h have the same length and contain the factor $\diamond_0 w_1\diamond_1 \ldots \diamond_{h-1}$, it follows from Lemma 1 that $\|S_h\| \leq \lfloor\frac{n-2h}{3}\rfloor + 1$. Therefore,

$$\|S\| \leq \lfloor\frac{n-2h}{3}\rfloor + 1 + \lfloor\frac{n-2h+5}{3}\rfloor \leq 2\lfloor\frac{n-2h}{3}\rfloor + 3$$

The upper bound for $\|S\|$ reached in the second case is always greater than or equal to the upper bound reached in the first case. Therefore,

$$\|S\| \leq 2\lfloor\frac{n-2h}{3}\rfloor + 3$$

□

Proposition 4. *The leading term in $m_{3,k}(n)$ is at most $\frac{3n}{4}k$.*

Proof. Let $w = w_0 \diamond w_1 \diamond w_2 \diamond w_3$ be a partial word of length n with three holes. We wish to count the number of possible factors of w that are squares containing two free holes. Let S_1 be all such factors wherein the first hole of w is *not* free. Define S_2 and S_3 similarly. We wish to find the size of $S = \cup_{1 \leq i \leq 3} S_i$. The types of factors in S_1, S_2, and S_3 are illustrated below (the first half of each factor is written above the second half to show the alignment of the holes):

S_1	$(w_0 \diamond w_1)''$	$\diamond$	w_2'
	w_2''	$\diamond$	w_3'
S_2	w_0''	$\diamond$	$(w_1 \diamond w_2)'$
	$(w_1 \diamond w_2)''$	$\diamond$	w_3'
S_3	w_0''	$\diamond$	w_1'
	w_1''	$\diamond$	$(w_2 \diamond w_3)'$

where v' and v'' denote a prefix and suffix of a word v respectively. Because all factors in S_1 have the second and third holes of w aligned, all factors in S_1 have the same length. Therefore, each factor in S_1 ends at a different position of $\diamond w_3$. Also, the first element of the second half of each factor in S_1 occurs at a different position of $w_2 \diamond$. Therefore, $\|S_1\| \leq |w_3| + 1$ and $\|S_1\| \leq |w_2| + 1$. We can use similar reasoning to arrive at the following relations:

$$\|S_1\| \leq |w_2| + 1 \qquad \|S_2\| \leq |w_0| + 1 \qquad \|S_3\| \leq |w_0| + 1$$
$$\|S_1\| \leq |w_3| + 1 \qquad \|S_2\| \leq |w_3| + 1 \qquad \|S_3\| \leq |w_1| + 1$$

Because $\|S\| = \|S_1\| + \|S_2\| + \|S_3\|$ and $n = |w_0| + |w_1| + |w_2| + |w_3| + 3$, we determine that

$$\|S\| \leq |w_2| + 1 + |w_3| + 1 + |w_1| + 1 = n - |w_0|$$
$$\|S\| \leq |w_2| + 1 + |w_3| + 1 + |w_0| + 1 = n - |w_1|$$
$$\|S\| \leq |w_3| + 1 + |w_0| + 1 + |w_1| + 1 = n - |w_2|$$
$$\|S\| \leq |w_2| + 1 + |w_0| + 1 + |w_1| + 1 = n - |w_3|$$

Therefore,

$$\|S\| \leq n - \max\{|w_0|, |w_1|, |w_2|, |w_3|\} \leq n - \lceil \frac{n-3}{4} \rceil \leq \frac{3n}{4}$$

□

As we show next, we can improve the bound for the case when there are only two holes present in the word.

Proposition 5. *If* $n \equiv 2 \bmod 6$, *then*

$$m_{2,k}(n) - \frac{n+1}{3}k \geq \frac{n-2}{2}$$

Proof. Using Theorem 1 and the fact that $n \equiv 2 \bmod 6$, the leading term in $m_{2,k}(n)$ is $\frac{n+1}{3}k$. Therefore, $m_{2,k}(n) - \frac{n+1}{3}k$ is the constant term of the polynomial $m_{2,k}(n)$. It suffices to construct a partial word w with two holes over a k-letter alphabet A with $|w| = n \equiv 2 \bmod 6$ such that w contains $\frac{n+1}{3}k + \frac{n-2}{2}$ distinct squares. Consider the word

$$w = (ab)^l \diamond (ab)^l \diamond (ab)^l$$

of length n over A, such that a, b are distinct letters of A with $l = \frac{n-2}{6}$. The following is an exhaustive list of general forms of factors of w that are squares:

$$\begin{array}{llll} (ab)^l\diamond, & b(ab)^{l-1}\diamond a, & \ldots, & \diamond(ab)^l \\ ab, & (ab)^2, & \ldots, & (ab)^{\lfloor \frac{l}{2} \rfloor} \\ ba, & (ba)^2, & \ldots, & (ba)^{\lceil \frac{l}{2} \rceil} \\ (ab)^0 a, & (ab)^1 a, & \ldots, & (ab)^{l-1}a \\ (ba)^0 b, & (ba)^1 b, & \ldots, & (ba)^{l-1}b \end{array}$$

Figure 1 illustrates these squares for $n = 32$. These general forms are pairwise incompatible. Thus, there are a total of

$$(2l+1)k + \lfloor \frac{l}{2} \rfloor + \lceil \frac{l}{2} \rceil + l + l = (\frac{n-2}{3} + 1)k + 3l = \frac{n+1}{3}k + \frac{n-2}{2}$$

distinct full words that are squares compatible with factors of w. □

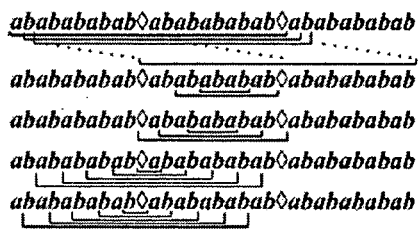


Figure 1: Squares in $(ab)^5 \diamond (ab)^5 \diamond (ab)^5$

3 Counting distinct squares: A second approach

At each position in a full word there are at most two distinct squares whose last occurrence starts, and thus the number of distinct squares in a word of length n is bounded by $2n$ as stated in the following theorem.

Theorem 2. [4] *Any full word of length n has at most $2n$ distinct squares.*

A short proof of Theorem 2 is given in [5]. It follows from the unique decomposition of words into primitive ones, and synchronization (a word w is primitive if and only if in ww there exist exactly two factors equal to w, namely the prefix and the suffix).

We now consider the one-hole case which behaves very differently from the zero-hole case. We will also count each square at the position where its last occurrence starts. If the last occurrence of a square in a partial word starts at position i, then it is a *square at position i*. In the case of partial words with one hole, there may be more than two squares that have their last occurrence starting at the same position. Such is the case with $a\diamond aababaab$ that has three squares at position 0: $a\diamond aa$, $a\diamond aaba$ and $a\diamond aababaab$. We will prove that if there are more than two squares at some position, then the hole is in the shortest square. We will also construct for $k \geq 2$, a partial word with one hole over a k-letter alphabet that has more than k squares at position 0. But first, we recall some results that will be useful for our purposes.

Lemma 2. [1] *Let $x, y \in A_\diamond^*$ be such that xy has at most one hole. If $xy \uparrow yx$, then there exist $z \in A^*$ and integers m, n such that $x \subset z^m$ and $y \subset z^n$.*

Lemma 3. [6] *Let $w \in A^*$. If $w = z_1z_2z_3 = z_2z_3z_4 = z_3z_4z_5$ for some $z_i \in A^* \setminus \{\varepsilon\}$, then there exist $x \in A^*$ primitive and integers p, q and r, $1 \leq p \leq r < q$, such that $x = x'x''$ for some $x' \in A^*$ and $x'' \in A^* \setminus \{\varepsilon\}$, and $z_1 = x^p$, $z_2 = x^{q-r}$, $z_3 = x^{r-p}x'$, $z_4 = x''x^{p-1}x'$, and $z_5 = x''x^{q-r-1}x'$.*

Theorem 3. *If a partial word with one hole has at least three distinct squares at the same position, then the hole is in the shortest square.*

Proof. Let uu', vv' and ww' be the three shortest squares whose last occurrence start at the same position, and assume that $|w| < |v| < |u|$. It is impossible for these three squares to be all full (otherwise the subword u^2, a full word, would have three squares starting at its position 0).

For a contradiction, let us assume that ww' is full (here $w = w'$). If $w^2 \leq u$, then the prefix of length $|w^2|$ of u' is a later occurrence of a square compatible with w^2. And so we must have $v < u < w^2$. If the hole is in u' but not in v', then $v = v'$, and by replacing the hole with the corresponding letter in u, we obtain the full word u^2 that has three distinct squares at position 0, a contradiction. If the hole is in v', then set $w^2 = uz_3$, $u = vz_2$ and $v = wz_1$. We get $w = z_1z_2z_3$, $v = z_1z_2z_3z_1$ and $u = z_1z_2z_3z_1z_2$. Let w_2 and w_3 be the prefixes of length $|w|$ of v' and u' respectively. Since z_2z_3 is a prefix of both v and v', let z_4 be such that $w, w_2 \subset z_2z_3z_4$. Note that $|z_4| = |z_1|$. Two cases occur.

Case 1. The hole is in the suffix of length $|v| - |w|$ of v'.

In this case, let z_5 be such that $w = z_3z_4z_5$. Note that $|z_5| = |z_2|$. Here $w = z_1z_2z_3 = z_2z_3z_4 = z_3z_4z_5$ and by Lemma 3, there exist $x \in A^*$ primitive and integers p, q and r, $1 \leq p \leq r < q$, such that $x = x'x''$ for some $x' \in A^*$, $x'' \in A^* \setminus \{\varepsilon\}$, and $z_1 = x^p$, $z_2 = x^{q-r}$, $z_3 = x^{r-p}x'$, $z_4 = x''x^{p-1}x'$, and $z_5 = x''x^{q-r-1}x'$.

We have $w = z_1z_2z_3 = x^qx'$, $v = wz_1 = x^qx'x^p$ and $u = vz_2 = x^qx'x^px^{q-r}$. If $x' = \varepsilon$, then a later occurrence of a square compatible with w^2 exists, and so we assume that $x' \neq \varepsilon$. Since the hole is in the suffix of length $|v| - |w|$ of v', the hole is in the suffix of length $|x^p|$ of v'. We can write $v' = x^qx'x^sx_1x_2x^{p-s-1}$ where $0 \leq s < p$, $|x_1| = |x'|$ and $|x_2| = |x''|$, and where the hole is in x_1 or x_2. Since $u \uparrow u'$, we have $z_1z_2z_3z_1z_2 \uparrow z_3z_4x^sx_1x_2x^{p-s-1}\ldots$, or $x^qx'x^px^{q-r} \uparrow x^rx'x^sx_1x_2x^{p-s-1}\ldots$. The fact that $r < q$ implies that $x^{q-r}x'x^px^{q-r} \uparrow x'x^sx_1x_2x^{p-s-1}\ldots$. If $s > 0$, then $x'x''x' = x'x'x''$ and $x''x' = x'x''$, and the latter being an equation of commutativity implies that a word y exists such that $x' = y^m$ and $x'' = y^n$ for some integers m, n. In this case, there is obviously a later occurrence of a square compatible with w^2. If $s = 0$, then $x^{q-r}x'x^px^{q-r} \uparrow x'x_1x_2x^{p-1}\ldots$. Since $q > r$, by looking at the prefixes of length $|xx'|$ we get $x'x''x' \uparrow x'x_1x_2$ and deduce $x''x' \uparrow x_1x_2$.

If the hole is in x_1, then $x_2 = x''$ and $x''x' \uparrow x_1x''$. By weakening, we get $x''x_1 \uparrow x_1x''$, an equation of commutativity that satisfies the conditions of Lemma 2 since $x''x_1$ has only one hole. Similarly as above, a word y exists such that $x_1 \subset y^m$ and $x'' = y^n$ for some integers m, n. Set $x_1 = y^ty'y^{m-t-1}$ where $0 \leq t < m$ and y' is the factor that contains the hole. Since $x_1 \subset x'$, we deduce that $x' = y^ty''y^{m-t-1}$ for some y''. The compatibility $x''x' \uparrow x_1x''$ implies $y^ny^ty''y^{m-t-1} \uparrow y^ty'y^{m-t-1}y^n$ and by simplification $y^ny'' \uparrow y'y^n$. Since $x'' \neq \varepsilon$, we have $n > 0$ and obtain $y'' = y$. We get $x' = y^m$, and there is obviously a later occurrence of a square compatible with w^2. We argue similarly in the case where the hole is in x_2.

Case 2. The hole is not in the suffix of length $|v| - |w|$ of v'.

In this case, set $w = z_2z_3z_4$ and $w_2 = z_2z_3z_4'$ and the hole is in z_4'. Also, set $w = z_3z_4''z_5$ and $w_3 = z_3z_4'z_5$ where both $z_4' \subset z_4$ and $z_4' \subset z_4''$, and $|z_5| = |z_2|$. We treat the case where $z_4'' \neq z_4$ and leave the case where $z_4'' = z_4$ to the reader.

If $z_4'' \neq z_4$, then put $z_1 = x^p$ where x is primitive and p is a positive integer. Since $z_1z_2z_3 = z_2z_3z_4$ and the equation $z_1(z_1z_2z_3) = (z_1z_2z_3)z_4$ is one of conjugacy, we can write $z_4 = x''x^{p-1}x'$, where $x = x'x''$ with x'' nonempty, and $z_1z_2z_3 = x^qx'$ for some $q \geq p$. Since $z_1z_2z_3 = x^qx'$ and $z_1 = x^p$, we have $z_2z_3 = x^{q-p}x'$. Say $z_2 = x^ty'$ where $t \geq 0$, and y' is a prefix of x with $y' \neq x$. Set $x = y'y''$ with y'' nonempty. If $y' = \varepsilon$, we have $z_2 = x^t$ and $z_3 = x^{q-p-t}x'$ and in this case $z_4'' = z_4$, a contradiction. This can be seen by using the equality $z_2z_3z_4 = z_3z_4''z_5$. And so $y' \neq \varepsilon$. Since z_4' has the length of z_1, write $z_4' = (x''x')^sx_2x_1(x''x')^{p-s-1}$ where $0 \leq s < p$, $|x_1| = |x'|$, $|x_2| = |x''|$, and where the hole is in x_1 or x_2. There are three cases to consider: (2.1) $t < q-p-1$; (2.2) $t = q-p-1$; and (2.3) $t = q-p$. We prove the second one, and leave the other two to the reader.

For (2.2), $z_2 = x^ty'$ and $z_3 = y''x'$. Since $z_1z_2z_3 = z_3z_4''z_5$, we have $x^qx' = y''x'\ldots$. We consider the case where $|x'| \geq |y'|$ and then the case where $|x'| < |y'|$. If $|x'| \geq |y'|$ or y' is a prefix of x', then since $q = p + t + 1 > 0$, the prefixes of length $|x|$ are $y'y''$ and $y''y'$ respectively and again, the equality $y'y'' = y''y'$ holds, and as above leads to a contradiction. If $|x'| < |y'|$ or x' is a prefix of y', then since $z_1z_2z_3 \uparrow z_3z_4'z_5$, we have $x^qx' \uparrow y''x'(x''x')^sx_2x_1(x''x')^{p-s-1}\ldots$.

If $s > 0$, then the fact that the prefixes of length $|x|$ are compatible implies that $y'y'' = y''y'$. If $s = 0$ and the hole is in x_1, then $x_2 = x''$ and $y''x'x'' = y''x = y''y'y''$ is a prefix of $z_3z_4'z_5$ in which case $y'y'' = y''y'$ as above. If $s = 0$ and the hole is in x_2, then $x_1 = x'$ and set $y' = x'y$ for some $y \neq \varepsilon$. Here, $x'' = yy''$, and put $x_2 = y_1y_2$ where $y_1 \subset y$ and $y_2 \subset y''$. We get $x^qx' \uparrow y''x'x_2x_1(x''x')^{p-1} \ldots = y''x'y_1y_2x'(x''x')^{p-1} \ldots$.

If the hole is in y_2, then $y_1 = y$ and $y''x'y_1 = y''x'y = y''y'$ is a prefix of $z_3z_4'z_5$ and the result again follows since $y'y'' = y''y'$. If the hole is in y_1, then $y'y'' \uparrow y''x'y_1$ or $x'yy'' \uparrow y''x'y_1$, and by weakening $(x'y_1)y'' \uparrow y''(x'y_1)$. The latter being an equation of commutativity, by Lemma 2, we get that $x'y_1 \subset z^m$ and $y'' = z^n$ for some word z and positive integers m, n. Set $x'y_1 = z^kz'z^{m-k-1}$ where $0 \leq k < m$ and z' is the factor that contains the hole. Since $x'y_1 \subset x'y$, we deduce that $x'y = z^kz''z^{m-k-1}$ for some z''. The compatibility $x'yy'' \uparrow y''x'y_1$ implies $z^kz''z^{m-k-1}z^n \uparrow z^nz^kz'z^{m-k-1}$. By simplification we obtain $z''z^n \uparrow z^nz'$, and since $n > 0$ we get $z'' = z$, and thus $y' = x'y = z^m$. The result follows since $x = y'y'' = z^{m+n}$ with $m + n > 1$. □

Proposition 6. *For $k \geq 2$, there exists a partial word with one hole over a k-letter alphabet that has more than k squares at position 0.*

Proof. Let $\Sigma = \{a_1, a_2, \ldots\}$ be an infinite ordered set. We build a sequence of partial words with one hole, $(DS_i)_{i\geq 2}$, where DS_i contains $i + 1$ squares with their last occurrence starting at position 0. In order to do this, we build an intermediary sequence of partial words with one hole $(DS_i')_{i\geq 2}$ and denote by $DS_i'(a)$, the word DS_i' in which the hole has been replaced by the letter a. Let $DS_2 = a_1\diamond a_1a_1a_2a_1a_2a_1a_1a_2$, and for $i \geq 3$,

$$\begin{aligned} DS_{i-1}' &= DS_{i-1}a_{i-1} \\ DS_i &= DS_{i-1}'DS_{i-1}'(a_i) \end{aligned}$$

In other words, DS_i consists of the concatenation of DS_{i-1} with the last letter of the smallest alphabet used for creating DS_{i-1}, concatenated again with the same factor in which the hole has been replaced by a letter not present in the word so far. For example,

$$\begin{aligned} DS_2' &= a_1\diamond a_1a_1a_2a_1a_2a_1a_1a_2a_2 \\ DS_3 &= a_1\diamond a_1a_1a_2a_1a_2a_1a_1a_2a_2a_1a_3a_1a_1a_2a_1a_2a_1a_1a_2a_2 \end{aligned}$$

the latter having three squares other than itself at position 0: $a_1\diamond a_1a_1a_2a_1a_2a_1a_1a_2$, $a_1\diamond a_1a_1$ and $a_1\diamond a_1a_1a_2a_1$. For $k \geq 2$, DS_k, a partial word with one hole over a k-letter alphabet, has $k+1$ squares. This is due to the fact that all previous squares cannot reappear later in the word because of the newly introduced letter. □

4 Conclusion

Although the computations done so far show that the actual bound for the one-hole partial words give us at most n distinct squares in any word of length n, the results obtained here using the approach of Fraenkel and Simpson make the bound directly dependable on the size of the alphabet. From our point of view, finding a dependency between the maximum number of squares starting at one position and the length of the word might be a solution. Solving this problem, at least partially, could also give a new perspective to the study of maximum distinct squares within a full word.

Note as well that for arbitrarily large alphabets of size k, we get an upper bound for all words containing h holes and having length n

$$g_{h,k}(n) \leq m_{h,k}(n) + k^{\lfloor \frac{h}{2} \rfloor}$$

This is due to the fact that the leading term is always maximal in $m_{h,k}$, hence adding one to its coefficient we get an upper bound.

References

[1] Berstel, J., Boasson, L. Partial words and a theorem of Fine and Wilf. *Theoretical Computer Science*, 218:135–141, 1999.

[2] Blanchet-Sadri, F. Algorithmic Combinatorics on Partial Words. Chapman & Hall/CRC Press, 2007.

[3] Fraenkel, A.S., Simpson, J. How many squares must a binary sequence contain? *Electronic Journal of Combinatorics*, 2(339 #R2): 1995.

[4] Fraenkel, A.S., Simpson, J. How many squares can a string contain? *Journal of Combinatorial Theory, Series A*, 82:112-120, 1998.

[5] Ilie, L. A simple proof that a word of length n has at most $2n$ distinct squares. *Journal of Combinatorial Theory, Series A*, 112:163-164, 2005.

[6] Ilie, L. A note on the number of squares in a word. *Theoretical Computer Science*, 380:373–376, 2007.

[7] Lothaire, M. Combinatorics on Words. Cambridge University Press, 1997.

[8] Smyth, W.F. Computing Patterns in Strings. Pearson Addison-Wesley, 2003.

Received 16th September 2008

Acta Cybernetica **19** (2009) 479–497.

Alphabetical Satisfiability Problem for Trace Equations*

L. Breveglieri†, A. Cherubini‡, C. Nuccio‡ and E. Rodaro§

Abstract

It is known that the satisfiability problem for equations over free partially commutative monoids is decidable but computationally hard. In this paper we consider the satisfiability problem for equations over free partially commutative monoids under the constraint that the solution is a subset of the alphabet. We prove that this problem is NP-complete for quadratic equations and that its uniform version is NP-complete for linear equations.

Keywords: free partially commutative monoid, trace equation, NP-complete problem

1 Introduction

The theory of word equations is an important subfield of the combinatorics on words firstly introduced in 1954 by Markov [9] who, given an alphabet Σ of constants, a set of unknowns Ξ and a word equation $W_L = W_R$ with $W_L, W_R \in (\Sigma \cup \Xi)^*$ proposed the problem of stating whether an assignment $\varphi : \Xi \to \Sigma^*$ exists such that $\varphi(W_L) = \varphi(W_R)$. This problem was solved more than 20 years later by Makanin [8] who gave a very complicated algorithm to decide whether or not a word equation with constants has a solution. Later several authors considered the problem of satisfiability of equations by a solution $\{\varphi(x) |\ x \in \Xi\}$ satisfying some constraints. In particular Robson and Diekert considered in [12] the problem of determining whether equations on free monoids have or not a solution with fixed lengths and they gave a linear algorithm for solving this problem for quadratic equations.
In the second half of '90 attention was paid also to equations on free partially commutative monoids. Free partially commutative monoids, firstly introduced in

*This work was partially supported by INDAM group GNSAGA, PRIN "Aspetti matematici e applicazioni emergenti degli automi e dei linguaggi formali" and ESF project AutoMathA.

†Politecnico di Milano, Department of Electronics and Information, Piazza L. da Vinci, 32, 20133 Milano, Italy, E-mail: luca.breveglieri@polimi.it.

‡Politecnico di Milano, Department of Mathematics, Piazza L. da Vinci, 32, 20133 Milano, Italy, E-mail: {alessandra.cherubini,claudia.nuccio}@polimi.it.

§Universitá dell'Insubria, DSCPI, Via Valleggio 11, 22100 Como, Italy, E-mail: emanuele.rodaro@gmail.com.

combinatorics [3], became very important in computer science for the theory of concurrence in connection with the semantics of labelled Petri nets [11] and the investigation of parallel program schemata [7]. The decidability of the satisfiability problem for equations on free partially commutative monoids, trace equations for short, was proved by Matiyasevich in [10] and by Diekert and al. [4, 5].
In this paper we consider the *alphabetical satisfiability problem* for trace equations with constants and unknowns, i.e. we look for the existence of a solution $\{\varphi(x) \in \Sigma | x \in \Xi\}$. The alphabetical satisfiability problem for trace equations presents some motivations coming from molecular biology and from reconstruction of sentences in natural languages. For instance, recently much attention was paid to partial words in the sense of [2], with motivations coming from different areas. Several of these motivations suggest that also *partial traces*, i.e. the generalization of partial words to trace monoids, deserve some attention. The alphabetical satisfiability problem is the generalization of the compatibility problem of partial words to the case of partial traces.
Using an argument which closely follows the proof of Theorem 1 in [6], we prove that the general problem of alphabetical satisfiability for quadratic word equations over a given free partially commutative monoid is NP-complete. Then we look for the complexity class of the alphabetical satisfiability problem for linear trace equations and we prove that the general problem is polynomial under particular assumptions on the independence alphabet while the uniform problem (i.e. the problem where even the independence alphabet is considered as variable parameter) is NP-complete.
In Section 2 we start giving some necessary notations and definitions, Section 3 shows that the general problem of alphabetical satisfiability for quadratic trace equations is NP-complete, while Sections 4 and 5 deal with the alphabetical satisfiability for linear trace equations.

2 Preliminaries

Let Σ be a finite alphabet and let $I \subseteq \Sigma \times \Sigma$ be a binary irreflexive and symmetric relation, called *independence* relation. We denote by $D = (\Sigma \times \Sigma) \setminus I$ the *dependence* relation, and by $\sim_I$ the least congruence over Σ^* generated by the relations $ab = ba$, for all $(a, b) \in I$. The pairs (Σ, I) and (Σ, D) are called, respectively, *independence* and *dependence alphabet*. For a subset A of Σ, let $I_A = (A \times A) \cap I$. If $I_A = \emptyset$, then A is called a *clique* of the dependence alphabet, or a D-clique. If $I_A = A \times A$, then A is called a *clique* of the independence alphabet, or a I-clique. The *free partially commutative* monoid (or trace monoid) over (Σ, I), is the quotient $\mathbb{M}(\Sigma, I) = \Sigma^*/\sim_I$ and it can be also denoted by $\mathbb{M}$, when no confusion arises. The elements of $\mathbb{M}$ are called *traces* and the trace with representative $x \in \Sigma^*$ is denoted by $[x]$.
Let Ξ be a finite set of unknowns and $\Theta = \Sigma \cup \Xi$. A *trace equation* with constants over (Σ, I) has the form $W_L \equiv W_R$ with $W_L, W_R \in \Theta^+$. A trace equation

$W_L \equiv W_R$ is called *linear* if each unknown occurs at most once in $W_L W_R$ and it is called *quadratic* if each unknown occurs at most twice in $W_L W_R$.
An *assignment* is a map $\varphi : \Xi \to \Sigma^*$. It can be extended to the monoid homomorphism $\varphi^* : \Theta^* \to \Sigma^*$ by putting $\varphi(a) = a$ for all $a \in \Sigma$. We say that the trace equation $W_L \equiv W_R$ is *satisfiable* if $\varphi^*(W_L) \sim_I \varphi^*(W_R)$ for some assignment φ. In such case we say also that φ satisfies $W_L \equiv W_R$ and the set $\{\varphi(x) \mid x \in \Xi\}$ is called a *solution* of $W_L \equiv W_R$. In the sequel, for simplicity, φ^* is still denoted by φ.
We say that the trace equation $W_L \equiv W_R$ is *alphabetically satisfiable* if it is satisfied by an assignment $\varphi : \Xi \to \Sigma$, then φ is called an *alphabetical assignment* and $\{\varphi(x) \mid x \in \Xi\}$ an *alphabetical solution* of the trace equation.
We look for alphabetical solutions of $W_L \equiv W_R$. It is obvious that, if $|W_L| \neq |W_R|$, no assignment $\varphi : \Xi \to \Sigma$ satisfies $W_L \equiv W_R$, hence we always assume that $|W_L| = |W_R|$ and when we refer to an assignment, we always consider an alphabetical assignment, if it is not differently specified.

3 Alphabetical satisfiability for quadratic trace equations

In this section we prove that the general problem of checking whether a quadratic trace equation over a given trace monoid $\mathbb{M}(\Sigma, I)$ has an alphabetical solution is an NP-complete problem.
We recall the following well-known result:

Proposition 1. *Let $(\Sigma, D) = \bigcup_{i=1}^{k} (A_i, D_i)$ be a union of subalphabets with $I_i = (A_i \times A_i) \setminus D_i$, $\mathbb{M}_i = \mathbb{M}(A_i, I_i)$ and let $[\pi_i] : \mathbb{M}(\Sigma, I) \to \mathbb{M}_i$ be the canonical homomorphisms for all $i \in \{1, 2, \ldots k\}$.*
Then the map $\overline{\pi} : \mathbb{M}(\Sigma, I) \to \mathbb{M}_1 \times \ldots \times \mathbb{M}_k, \qquad t \mapsto (\pi_1(t), \ldots, \pi_k(t))$
is an injective (canonical) homomorphism.

Remark 1. If the sets A_i are D-cliques, Proposition 1 says that two traces are equal if and only if their projections on the cliques A_i are equal.

In order to use the above Remark 1 in the case of trace equations and for all $A \subseteq \Sigma$ such that $A \times A \subseteq D$, we define the homomorphism $\overline{\pi}_A : (\Sigma \cup \Xi)^* \to (A \cup \Xi)^*$ such that, for all $x \in \Sigma \cup \Xi$,

$$\overline{\pi}_A(x) = \begin{cases} \epsilon & \text{if } x \notin A \cup \Xi \\ x & \text{otherwise} \end{cases}$$

For any $w \in (\Sigma \cup \Xi)^*$, the image $\overline{\pi}_A(w)$ is called *A-projection* of w.
As a direct consequence of Proposition 1 we get the following result:

Lemma 1. *Let $A_1, \ldots, A_k$ be cliques of the dependence alphabet of the trace monoid $\mathbb{M}(\Sigma, I)$. Then the trace equation $W_L \equiv W_R$ has a alphabetical solution if and only if there exists a family of assignments $\{\varphi_i : \Xi \to A_i \cup \{\epsilon\} \mid i = 1, \ldots, k\}$ such that:*

1. *for each $i \in \{1, \ldots, k\}$, φ_i satisfies the equation $\overline{\pi}_{A_i}(W_L) = \overline{\pi}_{A_i}(W_R)$;*

2. *for all* $x \in \Xi$ *there exists* $i \in \{1, \ldots, k\}$ *such that* $\varphi_i(x) \in A_i$;

3. *for all* $i \in \{1, \ldots, k\}$, $x \in \Xi$, *if* $\varphi_i(x) \in A_i$ *then, for all* $l \in \{1, \ldots, k\} \setminus \{i\}$, $\varphi_l(x) = \epsilon$ *if* $\varphi_i(x) \notin A_i \cap A_l$ *or* $\varphi_i(x) = \varphi_l(x)$ *if* $\varphi_i(x) \in A_i \cap A_l$.

We recall that a system of word equations is quadratic if each unknown occurs at most twice in the system. We use the following lemma whose proof is very close to the proof of Theorem 1 in [6]:

Lemma 2. *Let* $|\Sigma| \geq 2$. *The following problem is NP-complete.*
INSTANCE *A system of quadratic word equations.*
QUESTION *Is there an assignment* $\varphi : \Xi \to \Sigma \cup \{\epsilon\}$ *that satisfies the system?*

Proof. It is easy to verify that the problem is in NP. To prove that it is NP-hard we give a reduction from 3-SAT. Let $\mathcal{F} = C_0 \wedge C_1 \wedge \ldots \wedge C_{M-1}$ be a Boolean formula in 3-CNF over a finite set of variables Γ. Each clause has the form $C_i = l_{3i} \vee l_{3i+1} \vee l_{3i+2}$ where l_{3i+h} denotes a literal. We can assume that each variable has both positive and negative occurrences.
We associate the formula $\mathcal{F}$ with the following quadratic system $S(F, \Xi)$ of word equations with constants $a, b, a \neq b$ and the following set Ξ of unknowns:

- y_i, t_i, $0 \leq i \leq M-1$,
- x_j, $0 \leq j \leq 3M-1$,
- z_X, u_X, for all $X \in \Gamma$,
- $v_{X,s}$, for all $X \in \Gamma$, $0 < s \leq M-1$.

For each clause C_i we consider the equation

$$x_{3i}x_{3i+1}x_{3i+2} = ay_it_i \tag{1}$$

Now let $X \in \Gamma$ and consider the set of positions $D(X) = \{i_1, i_2, \ldots, i_r\}$ of the literal X in $\mathcal{F}$ and the set of positions $C(X) = \{j_1, j_2, \ldots, j_k\}$ of the literal $\neg X$ in $\mathcal{F}$. For each $X \in \Gamma$ we introduce another equation

$$L(X) = R(X) \tag{2}$$

where, if $r \leq k$,

$$L(X) = x_{i_1}x_{i_2} \ldots x_{i_r}v_{X,1}v_{X,2} \ldots v_{X,k-r}u_X a^k b x_{j_1}x_{j_2} \ldots x_{j_k}z_X$$

and

$$R(X) = a^k b a^k b$$

or, if $r > k$,

$$L(X) = x_{i_1}x_{i_2} \ldots x_{i_r}u_X a^r b x_{j_1}x_{j_2} \ldots x_{j_k}v_{X,1}v_{X,2} \ldots v_{X,r-k}z_X$$

and

$$R(X) = a^r b a^r b.$$

It is easy to see that $\mathcal{F}$ is satisfiable if and only if the system $S(F, \Xi)$, formed by the above equations (1),(2), has a solution whose lengths are not greater than 1. In fact if $\varphi : \Xi \to \Sigma \cup \{\epsilon\}$ is a possible assignment which satisfies the system then, encoding the value TRUE for the variable X of $\mathcal{F}$ in the fact that $\varphi(x_i) = a$ for all $i \in D(X)$, $\varphi(x_j) = \epsilon$ for all $j \in C(X)$ and the value FALSE in the fact that $\varphi(x_j) = a$ for all $j \in C(X)$, $\varphi(x_i) = \epsilon$ for all $i \in D(X)$, the equations of the form (1) guarantee that at least one literal in each clause assumes the value TRUE, while the equations of type (2) guarantee that the values of the literals are given in a coherent way. □

The next example illustrates the construction done in the proof of Lemma 1.

Example 1. Let us consider the following Boolean formula in 3-CNF over the set of variable $\Gamma = \{X_1, X_2, X_3\}$:

$$\mathcal{F} = (X_1 \vee \neg X_2 \vee X_3) \wedge (X_1 \vee \neg X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_2 \vee X_3).$$

We build the quadratic system $S(F, \Xi)$ of word equations with constants a, b, $a \neq b$ and this set Ξ of unknowns:

- y_i, t_i, $0 \leq i \leq 2$
- x_j, $0 \leq j \leq 8$
- $v_{X_1,1}, v_{X_2,1}, v_{X_3,1}$, $v_{X_1,2}, v_{X_2,2}, v_{X_3,2}$
- u_{X_1}, z_{X_1}, u_{X_2}, z_{X_2}, u_{X_3}, z_{X_3}

where the unknowns y_i, t_i are associated with the i-th clause C_i, the unknown x_i to the i-th literal in $\mathcal{F}$ and the unknowns $u_{X_i}, z_{X_i}, v_{X_i,s}$ to the variable X_i.
We associate with the three clauses of $\mathcal{F}$ these three word equations:

$$\begin{aligned} x_0x_1x_2 &= ay_0t_0 \\ x_3x_4x_5 &= ay_1t_1 \\ x_6x_7x_8 &= ay_2t_2 \end{aligned}$$

Then we introduce a word equation for each variable X_i. For example, for the variable X_1 we build the following word equation:

$$x_0x_3u_{X_1}a^2bx_6v_{X_1,1}z_{X_1} = a^2ba^2b. \tag{3}$$

The left side of the equation is built as follows. The variable X_1 occurs in the first, forth and seventh literal so in the equation we use the unknowns x_0, x_3, x_6. The factor $u_{X_1}a^2b$, where the exponent of a is the maximum between the numbers of positive and negative occurrences of X_1, is a separator between the unknowns that encode the positive and negative occurrences of X_1. After the unknown x_6 (that corresponds to the negative occurrence of X_1) we put a number of unknowns $v_{X_1,s}$ equal to the difference between the number of positive and negative occurrences of X_1 and at the end we put the variable z_{X_1}. The right side of the word equation

is a^2ba^2b where again the exponent of a is the maximum between the number of positive and negative occurrences of X_1.
Similarly, we proceed for the variable X_2 but, since the number of positive occurrences of X_2 is smaller than the number of its negative occurrences, after the unknown x_7 (encoding the positive occurrence of X_2 in the eighth literal) we put a number of unknowns $v_{X_2,s}$ equal to the difference between the number of negative and positive occurrences of X_2. We obtain the following equation:

$$x_7 v_{X_2,1} u_{X_2} a^2 b x_1 x_4 z_{X_2} = a^2 b a^2 b$$

Analogously we build the word equation relative to X_3 and we obtain this quadratic system of word equations:

$$\begin{cases} x_0 x_1 x_2 = a y_0 t_0 \\ x_3 x_4 x_5 = a y_1 t_1 \\ x_6 x_7 x_8 = a y_2 t_2 \\ x_0 x_3 u_{X_1} a^2 b x_6 v_{X_1,1} u_{X_1} = a^2 b a^2 b \\ x_7 v_{X_2,1} u_{X_2} a^2 b x_1 x_4 z_{X_2} = a^2 b a^2 b \\ x_2 x_8 u_{X_3} a^2 b x_5 v_{X_3,1} z_{X_3} = a^2 b a^2 b \end{cases} \quad (3)$$

The formula $\mathcal{F}$ is satisfiable if and only if there exists an assignment $\varphi : \Xi \to \{a, b\} \cup \{\epsilon\}$ that satisfies the previous system. Suppose that such an assignment exists and consider the word equation (3). Since its right side is a^2ba^2b and the left side contains the factor a^2b, it follows that

$$\varphi(x_0) = \varphi(x_3) = a,\ \varphi(u_{X_1}) = b \quad \text{and} \quad \varphi(x_6 v_{X_1,1} z_{X_1}) = \epsilon \quad (4)$$

or

$$\varphi(x_0 x_3 u_{X_1}) = \epsilon \quad \text{and} \quad \varphi(x_6) = \varphi(v_{X_1,1}) = a,\ \varphi(z_{X_1}) = b \quad (5)$$

Notice that if we encode the value TRUE (resp. FALSE) for the variable X in the fact that $\varphi(x_i) = a$ (resp. $\varphi(x_i) = \epsilon$) for all indices $i \in D(X)$ and $\varphi(x_j) = \epsilon$ (resp. $\varphi(x_j) = a$) for all indices $j \in C(X)$, conditions (4) and (5) mean that the assignment of truth value to X_1 is coherent. A similar argument applies for the coherence in the truth assignments to X_2 and X_3.
Now consider the first three word equations of the system relative to the three clauses of $\mathcal{F}$. The presence of the letter a in the right side encodes that at least one literal in each clause takes the value TRUE and so each clause is satisfied. Hence a truth assignment satisfying $\mathcal{F}$ corresponds to the assignment φ satisfying the system.
Viceversa, it is easy to verify that to each truth assignment that satisfies $\mathcal{F}$, corresponds an assignment $\varphi : \Xi \to \{a, b\} \cup \{\epsilon\}$ satisfying the systems.

As a consequence of Lemma 2 we can prove the following result:

Theorem 1. *The general problem of alphabetical satisfiability of a quadratic trace equation is NP-complete.*

Proof. It is quite obvious that this problem is in NP. Then to check that it is NP-hard we consider an alphabet $\Sigma = \{a, b, c, \sharp\}$ with the dependence relation D whose maximal D-cliques are $\{a, b, \sharp\}, \{c, \sharp\}$ and the set of unknowns Ξ defined in Lemma 2 and we give a reduction from 3-SAT. Using the notation introduced in the proof of Lemma 2 we consider the following equation:

$$\prod_{i=0}^{M-1} (x_{3i}x_{3i+1}x_{3i+2}\sharp) \prod_{X\in\Gamma} (L(X)\sharp) = \prod_{i=0}^{M-1} (ay_it_i\sharp) \prod_{X\in\Gamma} (R(X)c^{l_X}\sharp) \tag{6}$$

Let $I = (\Sigma^* \times \Sigma^*) \setminus D$. By Lemma 1, this equation has an alphabetical solution in $\mathbb{M}(\Sigma, I)$ if and only if its projections on the D-cliques $A_1 = \{a, b, \sharp\}$ and $A_2 = \{c, \sharp\}$ are satisfied respectively by the assignments φ_1 and φ_2 as in Lemma 1. Moreover such an assignment φ_1 exists if and only if there exists an assignment $\varphi : \Xi \to \{a, b\} \cup \{\epsilon\}$ which satisfies the system $S(F, \Xi)$. Indeed, notice that, for each assignment φ_1 satisfying the A_1-projection of equation (6) that assigns $\sharp$ to the unknowns of a subset $\Upsilon \subseteq \Xi$, there exists an assignment $\varphi : \Xi \to \{a, b\} \cup \{\epsilon\}$ which satisfies the system $S(F, \Xi)$ and such that

$$\forall z \in \Xi \qquad \varphi(z) = \begin{cases} \epsilon & \text{if } z \in \Upsilon \\ \varphi_1(z) & \text{otherwise} \end{cases}$$

Finally notice that, for each assignment φ_1 satisfying the A_1-projection of equation (6), there always exists an assignment φ_2 satisfying the A_2-projection of equation (6) such that φ_1 and φ_2 fulfill conditions in Lemma 1. Whence to decide whether equation (6) has an alphabetical solution is equivalent to decide whether the system $S(F, \Xi)$ has a solution whose lengths are not greater than 1. Then the problem is NP-complete. □

Remark 2. If in the proof of Theorem 1 we replace each occurrence of the symbol $\sharp$ with the string $ba^{2M+1}b$ where M is the number of clauses in the formula $\mathcal{F}$, we obtain a new equation having $(\{a, b, c\}, \{(a, a), (b, b), (a, b), (b, a), (c, c)\})$ as dependence alphabet. With some minor changes we can prove that such equation is satisfiable if and only if the system $S(F, \Xi)$ has a solution whose lengths are not greater than 1. Then the general problem of the alphabetical satisfiability of a quadratic trace equation is NP-complete even when $|\Sigma| = 3$.

4 The uniform problem of alphabetical satisfiability for linear trace equations

Let $w \in (\Sigma \cup \Xi)^+$, the sets $\mathcal{L}(w) = \{\varphi(w) |\ \varphi : \Xi \to \Sigma\}$ and $[\mathcal{L}](w) = \{[v] | v \in \mathcal{L}(w)\}$ are called respectively the *language associated with* w and the *trace language associated with* w.

Let $W_L \equiv W_R$ be a linear trace equation on the free partially commutative monoid $\mathbb{M}(\Sigma, I)$. The equation is satisfied by an alphabetical assignment $\varphi : \Xi \to \Sigma$ if and only if the finite trace languages associated with W_L and W_R have non empty

intersection. Since the membership problem for a regular trace language (i.e. for a trace language $T = \{[v] \mid\mid v \in R\}$, where R is a regular language) can be solved in polynomial time with respect to the length of the input word [1], there is a naive algorithm for checking whether a trace equation has or not an alphabetical solution. Obviously this algorithm has exponential time complexity because, for all $w \in (\Sigma \cup \Xi)^+$, the number of words in $[\mathcal{L}](w)$ is exponential with respect to the number of unknowns occurring in w.
It is natural to ask whether the alphabetical satisfiability problem is or not NP-complete. In particular, we obtained the following result:

Theorem 2. *The uniform problem of the alphabetical satisfiability for linear trace equations is NP-complete.*

Proof. Again, the difficult part is to prove that the problem is NP-hard. We give a reduction from 3-SAT. Let $\mathcal{F} = C_0 \wedge C_1 \wedge \ldots \wedge C_{M-1}$ be a Boolean formula in 3-CNF over a set of n variables $\Gamma = \{X_1, \ldots, X_n\}$, where

$$\forall j \in \{0, 1, \ldots, M-1\} \qquad C_j = l_{3j} \vee l_{3j+1} \vee l_{3j+2}$$

and l_{3j+h}, with $0 \leq h \leq 2$, are literals. We define the alphabet $\Sigma_{\mathcal{F}}$ and the independence relation $I_{\mathcal{F}}$ in the following way:

$$\Sigma_{\mathcal{F}} = \bigcup_{\substack{i \in \{1,\ldots,n\} \\ j \in \{0,1,\ldots,M-1\}}} \{\, d_i^j,\, c_i^j,\, z_i^j,\, u_i^j,\, e,\, \bot_j,\, \sharp_j,\, t_j \,\}$$

$$\begin{aligned} I_{\mathcal{F}} = \Bigg(& \bigcup_{\substack{i,j \in \{1,\ldots,n\} \\ h,k \in \{0,1,\ldots,M-1\}}} \Bigg(\bigcup_{\substack{k \geq h \\ i \neq j}} \{(z_i^h, d_j^k), (u_i^h, c_j^k),\} \cup \\ & \bigcup_{k \geq h} \{(u_i^h, t_k), (u_i^h, d_j^k), (z_i^h, c_j^k), (z_i^h, t_k)\} \cup \\ & \bigcup_{h > k} \{(u_i^h, \bot_k), (u_i^h, \sharp_k), (z_i^h, \bot_k), (z_i^h, \sharp_k), (\bot_k, d_j^h), (\sharp_k, d_j^h), (\bot_k, c_j^h), (\sharp_k, c_j^h)\} \cup \\ & \bigcup_{h \neq k} \{(u_i^h, u_j^k), (z_i^h, z_j^k)\} \cup \{(u_i^h, z_j^k) \mid h \neq k, i \neq j\} \cup \{(\bot_h, t_k), (\sharp_h, t_k)\} \cup \\ & \{(u_i^h, e), (z_i^h, e), (\bot_k, e), (\sharp_k, e)\} \Bigg) \Bigg)^{sym} \end{aligned}$$

where, for a binary relation R on an alphabet Σ, R^{sym} denotes the least symmetric relation on Σ containing R. Now, starting from the formula $\mathcal{F}$, we build a trace equation with constants in $\Sigma_{\mathcal{F}}$ and set of unknowns $\Xi = \{y_i \mid i = 0, 1, \ldots, 4M-1\}$ such that its subsets $\{y_i \mid i = 0, 1, \ldots, 3M\}$ and $\{y_i \mid i = 3M+1, \ldots, 4M-1\}$ are in one to one correspondence respectively with the set of literals and with the set of clauses. In this equation the letters d_i^j and c_i^j encode the fact that the variable X_i

has respectively a positive or negative occurrence in the clause C_j. The letters z_i^j and u_i^j mean that the variable X_i assumes respectively the value FALSE or TRUE in the clause C_j. The letters e encode the fact that the truth values of some variables occurring in a clause C are not relevant in order to satisfy the formula $\mathcal{F}$. In the sequel these variables of C and the unknowns associated with the literals of C where they occur are called *irrelevant variables* of C and *irrelevant unknowns* associated with C. The other variables and the unknowns associated with the literals where they occur are call *relevant variables* of C and *relevant unknowns* associated with C. Finally the letters $\perp_j, \sharp_j, t_j$ act like filters with the aim of assuring two conditions:

1. there is exactly one relevant unknown associated with each clause;
2. if y, y' are unknowns associated respectively with literals of two different clauses C_h and C_k where the same variable X_i occurs, then no assignment φ such that either $\varphi(y) = z_i^h$ and $\varphi(y') = u_i^k$ or $\varphi(y) = u_i^h$ and $\varphi(y') = z_i^k$ satisfies the equation.

This last condition corresponds to the fact that if a variable X_i is relevant in different clauses of $\mathcal{F}$ then the truth values assigned with X_i are coherent, i.e. X_i cannot assume the value TRUE in a clause and the value FALSE in another clause. For each $j \in \{0, 1, \ldots, M-1\}$ and $0 \leq h \leq 2$, we associate with the literal l_{3j+h} the unknown $y_{3j+h} \in \Xi$ and a letter $a_h^j \in \Sigma_{\mathcal{F}}$ where

$$a_h^j = \begin{cases} d_r^j & \text{if } l_{3j+h} = X_r \\ c_r^j & \text{if } l_{3j+h} = \neg X_r \end{cases}$$

In this way, each $C_j = l_{3j} \vee l_{3j+1} \vee l_{3j+2}$ is associated with the following word

$$w_j = y_{3j}\, y_{3j+1}\, y_{3j+2}\, a_0^j\, a_1^j\, a_2^j \in (\Sigma_{\mathcal{F}} \cup \Xi)^+.$$

Finally, we associate the formula $\mathcal{F}$ with the following words $w_{\mathcal{F}}, w'_{\mathcal{F}} \in (\Sigma_{\mathcal{F}} \cup \Xi)^+$:

$$w_{\mathcal{F}} = w_0 \sharp_0 \perp_0 w_1 \sharp_1 \perp_1 \ldots \sharp_{M-2} \perp_{M-2} w_{M-1} \sharp_{M-1} \perp_{M-1} t_{M-1} t_{M-2} \ldots t_1 t_0,$$

$$w'_{\mathcal{F}} = e^2 a_0 e^2 a_1 \ldots e^2 a_{M-1} t_{M-1} y_{3M} t_{M-2} y_{3M+1} \ldots t_0 y_{4M-1} \sharp \perp,$$

where, for all $j \in \{0, 1, \ldots, M-1\}$, $a_j = a_0^j a_1^j a_2^j$ and $\sharp\perp = \sharp_0 \perp_0 \sharp_1 \perp_1 \ldots \sharp_{M-1} \perp_{M-1}$. Then the formula $\mathcal{F}$ is satisfiable if and only if the trace equation $w_{\mathcal{F}} \equiv w'_{\mathcal{F}}$ has a solution.

We note that this equation is linear by the definition of the indices of the unknowns y and that it has polynomial size with respect to $\mathcal{F}$.

First, assume that the Boolean formula $\mathcal{F}$ is satisfiable, that is for each C_j there is at least one literal assuming the value TRUE. Pick for each C_j an $h_j \in \{0, 1, 2\}$ such that the literal l_{3j+h_j} assumes the value TRUE. Then consider the assignment $\varphi : \Xi \to \Sigma_{\mathcal{F}}$ defined in the following way: $\varphi(y_{3j+h_j}) = u_r^j$ if $l_{3j+h_j} = X_r$, $\varphi(y_{3j+h_j}) = z_r^j$ if $l_{3j+h_j} = \neg X_r$ and $\varphi(y_{3j+h}) = e$, for all $h \neq h_j$. For each $j = 0, 1, \ldots M-1$, put $\varphi(y_{4M-1-j}) = \varphi(y_{3j+h_j})$. Then from the definition of the independence relation it easily follows that

$$\varphi(w_{\mathcal{F}}) \sim_{I_{\mathcal{F}}} e^2 a_0 \ldots e^2 a_{M-1} t_{M-1} \varphi(y_{3M}) t_{M-2} \varphi(y_{3M+1}) \ldots t_0 \varphi(y_{4M-1}) \sharp \perp$$

and so the equation $w_{\mathcal{F}} \equiv w'_{\mathcal{F}}$ is satisfied by the assignment φ.
Conversely, assume that the equation $w_{\mathcal{F}} \equiv w'_{\mathcal{F}}$ is satisfied by an alphabetical assignment $\varphi : \Xi \to \Sigma_{\mathcal{F}}$ and prove that there exists an assignment of truth values to the variables of $\mathcal{F}$ which satisfies the formula.

Claim 1. The assignment φ has to assign the value e to at least $2M$ unknowns occurring in $w_{\mathcal{F}}$. Moreover, for each $j = 0, 1, \ldots, M-1$, two unknowns assigned to e have to occur in the words w_j, since the constants e are dependent on all the elements in the set $\{d_r^i, c_r^i |\ 0 \le i \le M-1, 1 \le r \le n\}$.

Claim 2. The remaining unknown occurring in w_j not assigned to e, say ξ_j, is such that $\varphi(\xi_j) \in \{z_r^s, u_r^s |\ 0 \le s \le j, 1 \le r \le n\}$. Indeed, $\varphi(\xi_j)$ has to take a value independent of e and of all the constants occurring in a_t for all $t \ge j$. But if $\varphi(\xi_j) \in \{\bot_s, \sharp_s | 0 \le s < j\}$ then $j > 0$ and so it is impossible to get the equivalence of $\varphi(w_{\mathcal{F}})$ with a word whose suffix is $\sharp\bot$.

Claim 3. Either $\varphi(\xi_j) = z_r^j$ when c_r^j occurs in a_j or $\varphi(\xi_j) = u_r^j$ when d_r^j occurs in a_j for some $r \in \{1, \ldots, n\}$. Indeed, if $s < j$ then $\varphi(\xi_j)$ and $\bot_{j-1}$ are dependent, hence $\sharp\bot$ cannot be the suffix of a word equivalent to $\varphi(w_{\mathcal{F}})$.

The occurrences of d_r^j or c_r^j in a_j indicate respectively that the literals X_r or $\neg X_r$ occur in C_j, then encoding with the letter u_r^j the value TRUE and with the letter z_r^j the value FALSE for the variable X_r in C_j, the assignment $\varphi(\xi_j)$ guarantees that at least one literal in C_j assumes the value TRUE. It remains to prove that each assignment satisfying the equation $w_{\mathcal{F}} \equiv w'_{\mathcal{F}}$ corresponds to a coherent way of assigning truth values to the variables in $\mathcal{F}$.

Claim 4. No $j, s \in \{0, 1, \ldots, M-1\}$ exist such that $\varphi(\xi_j) = u_r^j$ and $\varphi(\xi_s) = z_r^s$.
Indeed for each φ satisfying the equation we get

$$\varphi(w_{\mathcal{F}}) \sim_{I_{\mathcal{F}}} e^2 a_0 e^2 a_1 \ldots e^2 a_{M-1} \varphi(\xi_0)\varphi(\xi_1)\ldots\varphi(\xi_{M-1}) t_{M-1} \ldots t_1 t_0 \sharp\bot.$$

Then the assignment φ has to satisfy the trace equation

$$\xi_0\xi_1 \ldots \xi_{M-1} t_{M-1} \ldots t_1 t_0 \equiv t_{M-1} y_{3M} t_{M-2} y_{3M+1} \ldots t_0 y_{4M-1},$$

and this assures the claim 4. In fact suppose by contradiction that such j and s exist and that $j > s$. Then, since u_r^j and z_r^s are dependent and z_r^s depends on t_{s-1}, it follows that the assignment φ does not satisfy the last equation. So either the unknowns ξ_j, $j = 0, 1, \ldots, M-1$, which are not assigned to e correspond to different variables of $\mathcal{F}$, or if some of them correspond to the same variable X_r, φ gives them either the values u_r^j and u_r^s or the values z_r^j and z_r^s.
We can conclude that each assignment satisfying the equation encodes a truth assignment to the variables of $\mathcal{F}$ which satisfies the formula. □

The following example illustrates the construction done in the proof of Theorem 2, and also explains the rationale behind the definition of the alphabet and the independence relation.

Example 2. Let us consider the formula $\mathcal{F} = (X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_4)$.

Then we have:

$$\Sigma_{\mathcal{F}} = \bigcup_{\substack{i\in\{1,2,3,4\}\\ j\in\{0,1\}}} \{\, d_i^j,\, c_i^j,\, z_i^j,\, u_i^j,\, \bot_j,\, \sharp_j,\, t_j\ e\} \qquad \Xi = \{y_0, y_1, \ldots, y_7\}.$$

The left side and the right side of the trace equation associated with $\mathcal{F}$ are respectively:

$$w_{\mathcal{F}} = y_0\, y_1\, y_2\, d_1^0\, d_2^0\, c_3^0\, \sharp_0\, \bot_0\, y_3\, y_4\, y_5\, c_1^1\, c_2^1\, c_4^1\, \sharp_1\, \bot_1\, t_1\, t_0$$

$$w'_{\mathcal{F}} = e^2\, d_1^0\, d_2^0\, c_3^0\, e^2\, c_1^1\, c_2^1\, c_4^1\, t_1\, y_6\, t_0\, y_7\, \sharp_0\, \bot_0\, \sharp_1\, \bot_1\,.$$

We can associate with each coherent assignment of truth values the variables satisfying the formula $\mathcal{F}$ with an alphabetical assignment φ that satisfies the trace equation associated with $\mathcal{F}$. For instance, the formula $\mathcal{F}$ is satisfied giving the value TRUE to X_1 and the value FALSE to X_2, independently from the truth values assigned to the remaining variables. Then the relevant unknown associated with the first clause is y_0 and the relevant unknown associated with the second clause is y_4. It is easy to see that the assignment $\varphi : \Xi \to \Sigma_{\mathcal{F}}$ such that $\varphi(y_0) = \varphi(y_7) = u_1^0$, $\varphi(y_4) = \varphi(y_6) = z_2^1$, $\varphi(y_1) = \varphi(y_2) = \varphi(y_3) = \varphi(y_5) = e$ satisfies the trace equation $w_{\mathcal{F}} \equiv w'_{\mathcal{F}}$.

Conversely assume that there exists an alphabetical assignment φ that satisfies the trace equation $w_{\mathcal{F}} \equiv w'_{\mathcal{F}}$. Then at least four unknowns of $w_{\mathcal{F}}$ have to assume the value e. Only two unknowns in each set $\{y_0, y_1, y_2\}, \{y_3, y_4, y_5\}$ can take the value e because $(e, d_1^0), (e, c_1^1) \notin I_{\mathcal{F}}$. Then assume, for instance, $\varphi(y_1) = \varphi(y_2) = \varphi(y_4) = \varphi(y_5) = e$. Since $e^2\, d_1^0\, d_2^0\, c_3^0$ is a prefix of $\varphi(w'_{\mathcal{F}})$, $\varphi(y_0)$ has to be independent from the letters of that prefix, hence $\varphi(y_0) \in \{u_1^0, u_2^0, z_3^0\}$. In such case, by cancellativity, the assignment φ has to satisfy the following trace equation

$$\varphi(y_0)\sharp_0\bot_0\, y_3\, e^2\, c_1^1\, c_2^1\, c_4^1\, \sharp_1\, \bot_1\, t_1\, t_0 \equiv e^2\, c_1^1\, c_2^1\, c_4^1\, t_1\, y_6\, t_0\, y_7\, \sharp_0\, \bot_0\, \sharp_1\, \bot_1$$

whence $\varphi(y_3) \in \{z_1^1, z_2^1, z_4^1\} \cup \{\sharp_0, \bot_0\} \cup \{z_i^0, u_i^0 |\ 1 \le i \le 4\}$ and so

$$\varphi(y_0)\sharp_0\bot_0\, \varphi(y_3)\, e^2\, c_1^1\, c_2^1\, c_4^1\, \sharp_1\, \bot_1\, t_1\, t_0 \sim_{I_{\mathcal{F}}} e^2\, c_1^1\, c_2^1\, c_4^1\, \varphi(y_0)\sharp_0\bot_0\, \varphi(y_3) t_1\, t_0 \sharp_1\, \bot_1\,.$$

Using again cancellativity we obtain that the assignment φ has to satisfy the trace equation

$$\varphi(y_0)\sharp_0\bot_0\, \varphi(y_3)\, t_1\, t_0 \equiv t_1\, y_6\, t_0\, y_7\, \sharp_0\, \bot_0 \tag{7}$$

It is easy to deduce that $\{\varphi(y_6), \varphi(y_7)\} = \{\varphi(y_0), \varphi(y_3)\}$, i.e. the unknowns of the right side of the trace equation have always to assume the same values taken by the relevant unknowns of the left side.

Now we want to show that actually $\varphi(y_3) \in \{z_1^1, z_2^1, z_4^1\}$. By contradiction, let for instance $\varphi(y_3) = \sharp_0$, then we have $\varphi(y_0)\sharp_0\bot_0\, \varphi(y_3)\, t_1\, t_0 \sim_{I_{\mathcal{F}}} t_1 \varphi(y_0)\, t_0 \sharp_0\, \bot_0 \sharp_0$. So the trace equation (7) is not satisfied by φ because the word $\varphi(t_1\, y_6\, t_0\, y_7\, \sharp_0\, \bot_0)$ cannot be equivalent to a word whose suffix is $\bot_0\sharp_0$. Analogously we can prove that $\varphi(y_3) \neq \bot_0$.

Now suppose that $\varphi(y_3) \in \{z_i^0, u_i^0 |\ 1 \le i \le 4\}$ and, for instance, let $\varphi(y_3) = z_1^0$.

Then $\varphi(y_0)\sharp_0\perp_0\varphi(y_3)\,t_1\,t_0 \sim_{I_\mathcal{F}} t_1\,\varphi(y_0)\,t_0\sharp_0\perp_0 z_1^0$ and $(z_1^0, \perp_0) \notin I_\mathcal{F}$. It follows that the trace equation (7) is not satisfied by φ because $\varphi(t_1\,y_6\,t_0\,y_7\,\sharp_0\perp_0)$ cannot be equivalent to a word whose suffix is $\perp_0 z_1^0$.
We can deduce that $\varphi(y_3) \in \{z_1^1, z_2^1, z_4^1\}$ but notice that the choice of the values of $\varphi(y_0)$ and $\varphi(y_3)$ is tied, as required in the condition 2 in the proof of Theorem 2. If, for instance, $\varphi(y_0) = u_1^0$ then it is necessary that $\varphi(y_3) \neq z_1^1$. Indeed, if $\varphi(y_3) = z_1^1$ then $\varphi(y_0)\sharp_0\perp_0\varphi(y_3)\,t_1\,t_0 \sim_{I_\mathcal{F}} u_1^0 z_1^1 t_1\,t_0\sharp_0\perp_0$ hence, using cancellativity, by the equation (7) it follows that φ has to satisfy the trace equation $u_1^0 z_1^1 t_1\,t_0 \equiv t_1 y_6 t_0 y_7$. But this is impossible since $(u_1^0, z_1^1), (t_0, z_1^1) \notin I_\mathcal{F}$. This last fact allows to control the coherent assignment of the truth values to the relevant variables in $\mathcal{F}$. Indeed $\varphi(y_0) = u_1^0$ and $\varphi(y_3) = z_1^1$ would encode that X_1 is a relevant variable for both the first and the second clause and also that X_1 assumes in an incoherent way the value TRUE in the first clause and the value FALSE in the second one.
We can conclude that the assignment φ satisfying the trace equation encodes a coherent assignment of truth values to the variables of $\mathcal{F}$ which satisfies the formula.

5 Linear trace equations on free products of free commutative monoids

In this section we consider a linear trace equation $W_L \equiv W_R$ on a trace monoid $\mathbb{M}(\Sigma, I)$ that is a free product of free commutative monoids. This conditions means that the maximal I-cliques $C_1, \ldots, C_r$ of the independence alphabet (Σ, I) are disjoint. We give a polynomial time algorithm to solve the alphabetical satisfiability problem for such equation.

Remark 3. Let $[u], [v] \in \mathbb{M}$, and let $u = u_1 \ldots u_m$ be a decomposition of u such that, for each $i \in \{1, \ldots, m\}$, $u_i \in C_{j_i}^+$ for some $j_i \in \{1, \ldots, r\}$ and $j_i \neq j_{i+1}$. Then $[v] = [u]$ if and only if $v = v_1 \ldots v_m$ and, for each $i \in \{1, \ldots, m\}$, $v_i \sim_I u_i$.

In the sequel, for each word W we denote by $W(i)$ the i-th letter of W and by $W[i,k]$, $i \leq k$ the factor $W(i)W(i+1)\ldots W(k)$ of W.
As an immediate consequence of Remark 3, we obtain the following lemma:

Lemma 3. *Let φ be an alphabetical assignment that satisfies the trace equation $W_L \equiv W_R$ and let $i \in \{1, \ldots, |W_L|\}$, $k \in \{1, \ldots, r\}$ such that $W_L(i) = c \in C_k$ and $W_R(i) = x \in \Xi$ (resp. $W_L(i) = x \in \Xi$ and $W_R(i) = c \in C_k$). Then $\varphi(x) \in C_k$. Moreover there exist $1 = j_0 < j_1 < \ldots < j_s = |W_L| + 1$, such that $\varphi(W_L) = \prod_{i=0}^{s-1} \varphi(W_L[j_i, j_{i+1} - 1])$, $\varphi(W_R) = \prod_{i=0}^{s-1} \varphi(W_R[j_i, j_{i+1} - 1])$, with $\varphi(W_L[j_i, j_{i+1} - 1]) \in C_{k_i}^+$ for some $k_i \in \{1, \ldots, r\}$, $k_i \neq k_{i+1}$ and, for each $i \in \{0, \ldots, s-1\}$, $\varphi(W_L[j_i, j_{i+1} - 1]) \sim_I \varphi(W_R[j_i, j_{i+1} - 1])$.*

Now we introduce a function $\psi : \{0, 1, \ldots, |W_L| + 1\} \to \{-2, -1, 0, 1, \ldots, r\}$ to express when $W_L(i)$ and $W_R(i)$ belong to the same or to different maximal I-cliques

or if they are unknowns. This function is defined in the following way:

$$\forall i \in \{0,1,\ldots,|W_L|+1\} \qquad \psi(i) = \begin{cases} -2 & \text{if } i \in \{0,|W_L|+1\} \\ -1 & \text{if } W_L(i) \in C_h,\ W_R(i) \in C_k \\ & \text{with } h \neq k \\ 0 & \text{if } W_L(i), W_R(i) \in \Xi \\ h & \text{if } W_L(i), W_R(i) \in C_h \\ & \text{or } W_L(i) \in C_h \text{ and } W_R(i) \in \Xi \\ & \text{or } W_L(i) \in \Xi \text{ and } W_R(i) \in C_h \end{cases}$$

Definition 1. *A* microblock *associated with the trace equation $W_L \equiv W_R$ is a couple (i,j) of indices of $\{1,2,\ldots,n\}$ with $i \leq j$ such that*

1. $\forall k \in \{i,\ldots,j-1\} \qquad \psi(k) = \psi(k+1) \notin \{-1,-2\}$;
2. $\psi(i-1) \neq \psi(i)$ *and* $\psi(j) \neq \psi(j+1)$.

A microblock (i,j) is called microblock of unknowns *if $\psi(i) = 0$ and* microblock of type h *if $\psi(i) = h$.*

Notice that if a microblock (j,k) of unknowns is preceded and followed by two microblocks of the same type h then, without loss of generality, we can suppose that an alphabetical assignment that satisfies the trace equation assigns the unknowns of $W_L[j,k]$ and of $W_R[j,k]$ in the same I-clique C_h. An analog argument holds if the first or the final microblock are microblocks of unknowns. This fact is stated in the following lemma:

Lemma 4. *Let $(i,j-1),(j,k-1),(k,l)$ with $1 \leq i < j < k \leq l \leq |W_L|$ be microblocks associated with the linear trace equation $W_L \equiv W_R$ such that $(j,k-1)$ is a microblock of unknowns and $(i,j-1),(k,l)$ are microblocks of type h. Then the trace equation $W_L \equiv W_R$ is alphabetically satisfiable if and only if there exists an alphabetical assignment φ satisfying $W_L \equiv W_R$ such that $\varphi(W_L[j,k-1]), \varphi(W_R[j,k-1]) \in C_h^+$.*
Analogously, let $(1,k-1),(k,l)$ with $1 < k \leq l \leq |W_L|$ (resp. $(j,k-1),(k,|W_L|)$ with $1 \leq j < k \leq |W_L|$) be the first two (resp. the last two) microblocks associated with the linear trace equation $W_L \equiv W_R$ such that $(1,k-1)$, (resp. $(k,|W_L|)$) is a microblock of unknowns and (k,l), (resp. $(j,k-1)$) is a microblock of type h. Then the trace equation $W_L \equiv W_R$ is alphabetically satisfiable if and only if there exists an alphabetical assignment φ satisfying $W_L \equiv W_R$ such that $\varphi(W_L[1,k-1]) \in C_h^+$, (resp. $\varphi(W_L[k,|W_L|]) \in C_h^+$).

Proof. Let $W_L \equiv W_R$ be a satisfiable linear equation and let φ be an alphabetical assignment satisfying $W_L \equiv W_R$ such that $\varphi(W_L[j,k-1]) \notin C_h^+$ and $\varphi(W_L[j,k-1])$ contains the minimum number f of factors in C_t^+ for any $t \in \{1,\ldots,r\} \setminus \{h\}$. Suppose that $f \geq 1$ and let $s \in \{j,\ldots,k-1\}$ be the first index such that $\varphi(W_L[s,s+p]) \in C_t^+$ and $\varphi(W(s+p+1)) \notin C_t$ for some $t \in \{1,\ldots,r\}$, $t \neq h$ and for some natural number p with $s+p \leq k-1$. Then, by Lemma 3, $\varphi(W_R[s,s+p]) \in C_t^+$ and $\varphi(W_L[s,s+p]) \sim_I \varphi(W_R[s,s+p])$. Let ϕ be an alphabetical assignment such that

$\phi(W_L(q)) = \phi(W_R(q)) = z \in C_h$ for all $q \in \{s, \ldots, s+p\}$ and $\phi(x) = \varphi(x)$ for all $x \in \Xi$ that do not occur in $W_L[s, s+p]$ and $W_R[s, s+p]$. Then ϕ is an alphabetical assignment that satisfies $W_L \equiv W_R$ such that $\phi(W_L[j, k-1])$ contains less factors in C_t^+ for any $t \in \{1, \ldots, r\} \setminus \{h\}$ than $\varphi(W_L[j, k-1])$. This is a contradiction, hence $f = 0$. □

The previous lemma justifies the following definition:

Definition 2. *Let $h \in \{1, \ldots, r\}$. A* block of type h *associated with the linear trace equation $W_L \equiv W_R$ is a couple of indices (i, j), $i, j \in \{1, 2, \ldots, |W_L|\}$, $i \leq j$ such that there exist $i = i_1 < i_2 < \ldots < i_s = j + 1$ satisfying the following properties:*

1. $$W_L[i, j] = \prod_{1 \leq k < s} W_L[i_k, i_{k+1} - 1]$$

2. *there exists $h \in \{1, \ldots, r\}$ such that $(i, i_2 - 1), (i_{s-1}, i_s - 1)$ are microblocks of type h and, for all $k \in \{i, \ldots, j\}$, $\psi(k) \in \{0, h\}$.*

The factors $W_L[i, j]$ and $W_R[i, j]$ are called respectively left factor and right factor associated with the block (i, j). *Obviously each microblock is a block.*
A macroblock of type h *associated with the trace equation $W_L \equiv W_R$ is a couple of indices $i, j \in \{1, 2, \ldots, |W_L|\}$ with $i \leq j$ such that there exists a block (i', j') of type h satisfying the following properties:*

1. $i \leq i' \leq j' \leq j$;
2. *if $i \neq i'$ (resp. $j \neq j'$) then, for all $k \in \{i, \ldots, i'\}$ (resp. $k \in \{j', \ldots, j\}$), $\psi(k) = 0$.*

Now we describe a linear algorithm to state whether the linear trace equation $W_L \equiv W_R$ on the trace monoid on $\mathbb{M}(\Sigma, I)$ is satisfiable when the maximal I-cliques $C_1, \ldots, C_r$ of the independence alphabet (Σ, I) are disjoint. Roughly speaking, the algorithm works in the following way. It identifies the blocks associated with $W_L \equiv W_R$ and checks if $W_L[i, j] \equiv W_R[i, j]$ in each block (i, j). If so the equation is satisfiable. If for some block (i, j) the trace equation $W_L[i, j] \equiv W_R[i, j]$ is not satisfied and no block of unknowns is adjacent to (i, j), the algorithm exits and outputs "NO". Otherwise it extends the block (i, j) to a macroblock (i', j') including also these new unknowns and it checks whether $W_L[i', j'] \equiv W_R[i', j']$ is satisfiable. The procedure is described in Algorithm 1.
Now we check the correctness of the Algorithm 1. The procedure from line 1 to line 13 identifies the blocks associated with the trace equation by building an array η whose odd and even cells contain respectively the beginning and the end of a block of a certain type h with $h \neq 0$. The initial and final blocks identified by the previous procedure can be macroblocks. In fact if the trace equation begins or finishes with microblocks of unknowns, thanks to Lemma 4 we can assume without loss of generality that an alphabetical assignment satisfying the trace equation gives to the unknowns of these microblocks values in the same I-cliques of the adjacent

Algorithm 1 Equation satisfiability with disjoint maximal I-cliques

1: $\eta(1) \leftarrow 1$, $k \leftarrow 2$,$c \leftarrow 1$
2: **for** $i = 1$ to $|W_L|$ **do**
3: **if** $\psi(i) = -1$ **then**
4: Exit and write "NO"
5: **end if**
6: **if** $\psi(i) \neq 0$ **then**
7: **if** $\psi(c) \notin \{\psi(i), 0\}$ **then**
8: $\eta(k) \leftarrow c$, $\eta(k+1) \leftarrow i$, $k \leftarrow k+2$
9: **end if**
10: $c \leftarrow i$
11: **end if**
12: **end for**
13: $\eta(k) \leftarrow |W_L|$
14: $in \leftarrow 1$
15: **for** $t = 1$ to $k/2$ **do**
16: $out \leftarrow \eta(2t)$
17: **for** $i = 1$ to $|\Sigma|$ **do**
18: $K_R(i) \leftarrow 0$
19: $K_L(i) \leftarrow 0$
20: **end for**
21: **for** $i = in$ to out **do**
22: **for** $j = 1$ to $|\Sigma|$ **do**
23: **if** $W_L(i) = a(j)$ **then**
24: $K_L(j) \leftarrow K_L(j) + 1$
25: **end if**
26: **if** $W_R(i) = a(j)$ **then**
27: $K_R(j) \leftarrow K_R(j) + 1$
28: **end if**
29: **end for**
30: **end for**
31: $\theta \leftarrow 0$
32: **for** $j = 1$ to $|\Sigma|$ **do**
33: $H(j) \leftarrow K_R(j) - K_L(j)$
34: **if** $H(j) > 0$ **then**
35: $\theta \leftarrow \theta + H(j)$
36: **end if**
37: **end for**
38: $\lambda \leftarrow out - in + 1 - \sum_{s=1}^{|\Sigma|} K_L(s)$
39: **if** $\lambda - \theta \leq 0$ **then**
40: **if** $\eta(2t+1) - out - 1 < |\lambda - \theta|$ **then**
41: Exit and write "NO"
42: **else**
43: $in \leftarrow out + |\lambda - \theta| + 1$
44: **end if**
45: **else**
46: $in \leftarrow out + 1$
47: **end if**
48: **end for**
49: Exit and write "YES"

block. Hence we can directly consider the initial and final macroblocks.
Let (in, out) with $in = 1$ be the first block associated with the trace equation and suppose that it is of type l. By Lemmas 3 and 4 we can assume, without loss of generality, that an alphabetical assignment satisfying the trace equation gives to the unknowns some values in the same maximal I-clique C_l. So we can check if such an assignment exists just counting the number of occurrences of a letter $a(i) \in \Sigma$ in the left and right factors associated with (in, out). Hence the procedure from line 17 to line 30 defines two arrays K_L and K_R whose i-th cell contains respectively $|W_L[in, out]|_{a(i)}$ and $|W_R[in, out]|_{a(i)}$, i.e. the number of the occurrences of the letter $a(i)$ respectively in the left and in the right factor associated with the block (in, out). If $K_R(j) - K_L(j) > 0$ (resp. $K_R(j) - K_L(j) < 0$) for some j, it means that $a(j)$ is a *right surplus constant* (resp. *left surplus constant*), i.e. $a(j)$ has a number of occurrences in the right (resp. left) factor associated with (in, out) greater than in the left (resp. right) factor. Hence a suitable number of unknowns of $W_L[in, out]$ (resp. $W_R[in, out]$) has to be assigned to $a(j)$. Obviously if $K_R(j) - K_L(j) = 0$ we have no constraints on the assignments to the unknowns.
Now we have to check if the number of unknowns in the left (resp. right) factor associated with (in, out) is sufficient to match with the right (resp. left) surplus constants. Therefore we introduce the following notations:

- λ: number of unknowns in the left factor associated with (in, out);
- β: number of unknowns in the right factor associated with (in, out);
- θ: number of the right surplus constants of (in, out);
- ρ: number of the left surplus constants of (in, out).

So we have to consider the differences $\lambda - \theta$ and $\beta - \rho$. Using the arrays K_L and K_R we have:

$$\lambda = out - in + 1 - \sum_{s=1}^{|\Sigma|} K_L(s) \qquad \beta = out - in + 1 - \sum_{s=1}^{|\Sigma|} K_R(s)$$

$$\theta = \sum_{\substack{i \in \{1,\ldots,|\Sigma|\}, \\ K_R(i) > K_L(i)}} (K_R(i) - K_L(i)) \qquad \rho = \sum_{\substack{i \in \{1,\ldots,|\Sigma|\}, \\ K_L(i) > K_R(i)}} (K_L(i) - K_R(i))$$

A trivial verification shows that $\lambda - \theta = \beta - \rho$, hence the procedure from line 31 to 38 determines θ and λ. If $\lambda - \theta \leq 0$, it means that the number of unknowns in the left factor are not sufficient to match with the right surplus constants, hence the trace equation $W_L[in, out] \equiv W_R[in, out]$ is not alphabetically satisfiable. It follows that the initial trace equation $W_L \equiv W_R$ can be satisfied only if there exists a block $(out + 1, out + s)$ of unknowns such that $s \geq |\lambda - \theta|$. In that case the trace equation $W_L[in, out + |\lambda - \theta|] \equiv W_R[in, out + |\lambda - \theta|]$ is alphabetically satisfiable. If $|\lambda - \theta| < s$, the procedure from line 39 to line 47 adds the remaining $s - |\lambda - \theta|$ unknowns of the factors associated with $(out + 1, out + s)$ to the factors associated

with the block following $(out+1, out+s)$ obtaining a new macroblock. Then the entire process is iterated considering the successive macroblock (in, out).
Notice that the algorithm works in linear time. Clearly the procedure 1-14 require a linear time, hence let us consider the cycle of lines 15-48. It runs on the number $k/2$ of macroblocks associated with the trace equation. Let us denote by $(in(i), out(i))$ the macroblock in the i^{th} iteration and put $l_i = out(i) - in(i)$. Since the cycle of lines 21-30 scans the macroblock $(in(i), out(i))$ and $[in(i), out(i)] \cap [in(i+1), out(i+1)] = \emptyset$ (where, for all $p, q \in \mathbb{N}$ such that $p \leq q$, $[p, q]$ is the subset of natural numbers $\{p, p+1, \ldots, q\}$), we obtain that

$$\sum_{i=1}^{k/2} l_i \leq n.$$

Hence, an easy calculation allow to conclude that the number of steps in the cycle 15-48 is $O(n)$.

Example 3. Let $\Sigma = \{a, b, c, d\}$ and let $I = \{(a,b), (b,a), (c,d), (d,c)\}$ be the independence relation. Hence $C_1 = \{a, b\}$, $C_2 = \{c, d\}$ are the maximal I-cliques. Let us consider the trace equation $W_L \equiv W_R$, where

$$W_L = ax_1adx_2x_3ddcdddx_4x_5x_6a$$

and

$$W_R = bay_1cy_2y_3cdcy_4ccy_5y_6y_7b.$$

The division of the equation in blocks is the following:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	x_1	a	d	x_2	x_3	d	d	c	d	d	d	x_4	x_5	x_6	a
b	a	y_1	c	y_2	y_3	c	d	c	y_4	c	c	y_5	y_6	y_7	b

The array η relative to the beginning and the end of each block of type 1 or 2 is $\eta = (1, 3, 4, 12, 16, 16)$.
It is easy to see that $W_L[1,3] \equiv W_R[1,3]$ is alphabetically satisfiable.
In the next block $(4, 12)$ we have

$$K_L = (0,0,1,6),\ K_R = (0,0,5,1),\ H = (0,0,4,-5),\ \theta = 4,\ \lambda = 2$$

so $\lambda - \theta < 0$ and the trace equation $W_L[4,12] \equiv W_R[4,12]$ is not alphabetically satisfiable. But $\eta(5) - 12 - 1 \geq |\lambda - \theta|$ hence we can extend the block $(4,12)$ to the macroblock $(4,14)$ obtaining a trace equation $W_L[4,14] \equiv W_R[4,14]$ that is alphabetically satisfiable.
We include the remaining unknowns in position 15 to the factors associated with the next block $(16,16)$ that therefore becomes the macroblock $(15,16)$. It is evident that $W_L[15,16] \equiv W_R[15,16]$ is alphabetically satisfiable, so we can conclude that the initial trace equation $W_L \equiv W_R$ is alphabetically satisfiable too.

6 Conclusion

It is still an open problem to determine the complexity class of the general alphabetical satisfiability problem for a linear trace equation. The problem is polynomial when the number of unknowns in one side of the equation is logarithmic with respect the length of a member of the equation.
In Section 5 we proved that there is a polynomial time algorithm to check the alphabetical satisfiability problem on free products of free commutative monoids. We also have a (not yet published) polynomial time algorithm to solve the satisfiability problem for linear trace equations on free products with amalgamation of free commutative monoids, i.e. partially free commutative monoids whose maximal I-cliques $C_1, \ldots, C_r$ satisfy the condition $C_i \cap C_j = \bigcap_{k=1}^{r} C_k$ for all $i, j \in \{1, \ldots, r\}$. Previous monoids are both particular cases of free partially commutative monoids that are free products of free products with amalgamation of free commutative monoids, i.e. monoids whose independence alphabet (Σ, I) fulfils the following conditions: Σ is a disjoint union of Σ_i such that each I-clique of (Σ, I) is contained in some Σ_i and for each pair of I-cliques C_h,C_k contained in Σ_i for some i, $C_h \cap C_k$ is equal to the intersection of all the I-cliques contained in Σ_i. So we strongly conjecture that also in this case there is a polynomial algorithm for the alphabetical satisfiability problem for linear trace equations.
It would be interesting to consider in the future the complexity class of the alphabetical satisfiability problem for linear equations under some other constraints on the independence alphabet in order to have some hint for the general case.

References

[1] A. Bertoni, G. Mauri, N. Sabadini, *Equivalence and membership problems trace languages*, LNCS 140, 1982, 61-71.

[2] F. Blanchet-Sadri, *Algorithmic Combinatorics on Partial Words*, Chapman and Hall/CRC Press, Boca Raton, FL, 2008.

[3] P. Cartier and D. Foata, *Problèmes combinatoires de commutation and réarrangements*, LNM 85, 1969.

[4] V. Diekert, Yu. Matiyasevich and A. Muscholl, *Solving trace equations using lexicographical normal forms*, LNCS 1256, 1997, 336-347.

[5] V. Diekert, Yu. Matiyasevich and A. Muscholl, *Solving word equations modulo partial commutations*, TCS 224, 1999, 224-215.

[6] V. Diekert and J. Robson, *Quadratic word equations*, Jewels are Forever, 1999, 314-326.

[7] R. M. Keller, *Parallel program schemata and maximal parallelism I. Fundamental results*, Journal of A. C. M. 20 3, 1973, 514-537.

[8] G. S. Makanin, *The problem of solvability of equations in a free semigroup*, Math. Sbornik 103, 1977, 147-236, English transl. in Math. URSS Sbornik 32 (1977).

[9] A. A. Markov, *The theory of algorithms*, Trudy Math. Inst. Steklov. 42, (1954).

[10] Yu. Matiyasevich, *Some decision problems for traces*, LNCS 1234, 1997, 248-257.

[11] A. A. Mazurkiewicz, *Trace theory*, LNCS 255, 1987, 279-324.

[12] J. M. Robson and V. Diekert, *On quadratic word equations*, LNCS 1563, 1999, 217-226.

[13] V. Diekert and G. Rozenberg editors, *The book of traces*, World Scientific, 1995.

Received 12th October 2008

Acta Cybernetica 19 (2009) 499–516.

Homomorphisms Preserving Types of Density*

Helmut Jürgensen† and Ian McQuillan‡

Abstract

The concept of density in a free monoid can be generalized from the infix relation to arbitrary relations. Many of the properties known for density can be established over these more general notions of densities. In this paper, we investigate homomorphisms which preserve different types of density. We demonstrate a strict hierarchy between families of homomorphisms which preserve density over different types of relations. However, as with the case of endomorphisms, a similar hierarchy for weak-coding homomorphisms collapses. We also present an algorithm to decide whether a homomorphism preserves density over any relation which satisfies some natural conditions.
Keywords: density, homomorphisms, coding theory, formal language theory

1 Introduction

A language is *dense* if every word over the alphabet is the infix of some word in the language. One can use other relations ϱ in place of the infix relation to define other types of density. Then, a language could be ϱ-dense, with traditionally density being a special case where $\varrho = \leq_i$, the infix order. These types of densities arise naturally in the theory of codes (see [4]). Indeed, the notions of density, residues, ideals, closure, independence and maximality can be defined for arbitrary relations and properties can be established over these more general notions [3].

We will especially examine those relations which are important for the theory of codes. In particular, the length, prefix, suffix, infix, embedding, outfix, division, commutation and power relations. These produce the following families of codes, respectively, when examining independent sets [4, 5]: block codes (also called uniform codes), prefix codes, suffix codes, infix codes, hypercodes, outfix codes, 2-ps-codes, 2-codes and a superset of the 2-codes. Here, we will examine the same relations with respect to density.

*This research was funded in part by grants from the Natural Sciences and Engineering Research Council of Canada (H. Jürgensen and I. McQuillan).

†Department of Computer Science, University of Western Ontario, London, ON, N6A 5B7, Canada

‡Department of Computer Science, University of Saskatchewan, Saskatoon, SK, S7N 5A9, Canada, E-mail: mcquillan@cs.usask.ca

A homomorphism α from X^* to Y^* is said to preserve density if $\alpha(L)$ is dense over Y for every dense language L over X. For our purposes, we will say that a homomorphism α preserves ϱ-density if $\alpha(L)$ is ϱ-dense over Y for every ϱ-dense language L over X. The work in [3] mainly concerns determining which endomorphisms (homomorphisms where $X = Y$) preserve different types of densities. It is shown there that if ϱ is reflexive, transitive and compatible with homomorphisms (that is, $(x, y) \in \varrho$ implies $(\alpha(x), \alpha(y)) \in \varrho$) where $\varrho \subseteq \omega_n$, for some n (a large relation containing many standard relations), then a homomorphisms α preserves ϱ-density if and only if α restricted to X is a permutation of X. Many types of densities apply here including density defined with the prefix, suffix, infix, power, commutation and division relations. Hence, one gets "regular" density as a special case. In [3] it is left as an open problem to study the same problem over arbitrary homomorphisms, that is, homomorphisms where the alphabets X and Y can differ.

In this paper, we tackle the problem for weak-coding homomorphisms, and also for arbitrary homomorphisms. If ϱ is alphabet preserving ($(x, y) \in \varrho$ implies the set of letters of x is a subset of the letters of y), reflexive, and compatible with a homomorphism α from X^* to Y^*, then α preserves density if and only if every letter of Y appears in $\alpha(X)$. For arbitrary homomorphisms, the situation turns out to be quite a bit more complex. Indeed, the family of homomorphisms which preserves different types of densities forms a sometimes strict, sometimes collapsing hierarchy established in Theorem 3. The property used to separate, or collapse families in the hierarchy is given in Definition 8. This says that two relations ϱ_1 and ϱ_2 are *densely equivalent* if for every finite language L, L^* is ϱ_1-dense if and only if L^* is ϱ_2-dense. Then, Theorem 2 establishes that two relations which are transitive, reflexive and compatible with arbitrary homomorphisms are densely equivalent if and only if the families of homomorphisms preserving both types of density are identical. Hence, we can determine where the hierarchies collapse or are strict by deciding whether the two relations are densely equivalent. In addition, in Section 6 we show that we can decide if a given homomorphism preserves ϱ-density for any of prefix, suffix, infix, embedded, equality, division, commutation or power density.

2 Preliminaries and notation

In this section, we define the mathematical preliminaries necessary for this paper.

The symbol $\mathbb{N}$ denotes the set of positive integers and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For a set S, let $|S|$ denote the cardinality of S. Let S and T be sets and α a mapping of S into T. For a subset S' of S, $\alpha|_{S'}$ denotes the restriction of α to S'.

For a binary relation $\varrho \subseteq S \times T$, the set $\mathrm{dom}(\varrho) = \{s \mid s \in S, \exists t \in T, (s, t) \in \varrho\}$ is the *domain* of ϱ. Moreover, $\varrho^{-1} = \{(t, s) \mid (s, t) \in \varrho\}$ is the *inverse* of ϱ, and, for $s \in S$, $\varrho(s) = \{t \mid t \in T, (s, t) \in \varrho\}$. Consequently, $\varrho^{-1}(t) = \{s \mid s \in S, (s, t) \in \varrho\}$ for $t \in T$. In the sequel, for a binary relation $\varrho \subseteq S \times T, x \in S, y \in T$, we use, interchangeably, the notation $(x, y) \in \varrho$ and $x \; \varrho \; y$.

Let Γ be a countably infinite alphabet. In this paper, a finite alphabet will be any finite $X \subseteq \Gamma$. Furthermore, X and Y will be finite alphabets throughout

this paper. Then X^* is the set of all *words* over X including the *empty word* λ. Let $w \in X^*$ and $a \in X$, then $|w|_a$ is the number of occurrences of a in w and $|w| = \sum_{a \in X} |w|_a$ is the length of w. A language over X is a subset of X^* and a language is any set whereby there exists an alphabet X such that $L \subseteq X^*$. For a language L over X, alph(L) is the set of all $a \in X$ with $|w|_a > 0$ for some $w \in L$. For a word $w \in X^*$, we define the *reversal* of w, denoted w^R by $w^R = w$ if $w = \lambda$ and $w^R = a_n \cdots a_1$ if $w = a_1 \cdots a_n, n \geq 1; a_i \in X, 1 \leq i \leq n$.

Let X, Y be finite alphabets. Then a function α from X^* to Y^* such that $\alpha(xy) = \alpha(x)\alpha(y)$ for all $x, y \in X^*$ is called a homomorphism. A homomorphism $\alpha : X^* \to Y^*$ is called a coding if $|\alpha(a)| = 1$ for every $a \in X$. Also, α is called a weak coding if $|\alpha(a)| \leq 1$ for every $a \in X$. Moreover, α is called an endomorphism if $X = Y$.

Let $L, R \subseteq \Sigma^*$. We denote by $R^{-1}L = \{z \in \Sigma^* \mid yz \in L \text{ for some } y \in R\}$ and $LR^{-1} = \{z \in \Sigma^* \mid zy \in L \text{ for some } y \in R\}$.

3 Relations

We will use certain relations frequently which are important for code-related languages [5, 4].

Example 1. Let w and v be arbitrary words in Γ^*.

1. *Embedding order:* $w \leq_{\mathrm{e}} v$ if and only if there exist $n \in \mathbb{N}_0$, $w_1, \ldots, w_n$ and $v_0, v_1, \ldots, v_n$ in Γ^* such that $w = w_1 w_2 \cdots w_n$ and $v = v_0 w_1 v_1 w_2 \cdots w_n v_n$.
2. *Length order:* $w \leq_{\mathrm{u}} v$ if and only if $w = v$ or $|w| < |v|$.
3. *Prefix order:* $w \leq_{\mathrm{p}} v$ if and only if $v = wx$ for some $x \in \Gamma^*$.
4. *Suffix order:* $w \leq_{\mathrm{s}} v$ if and only if $v = xw$ for some $x \in \Gamma^*$.
5. *Outfix relation:* $w \leq_{\mathrm{o}} v$ if and only if there are $w_1, u, w_2 \in \Gamma^*$ such that $v = w_1 u w_2$ and $w = w_1 w_2$.
6. *Infix order:* $w \leq_{\mathrm{i}} v$ if and only if $v = xwy$ for some $x, y \in \Gamma^*$.
7. *Division order:* $w \leq_{\mathrm{d}} v$ if and only if $v = wx = yw$ for some $x, y \in \Gamma^*$.
8. *Commute order:* $w \leq_{\mathrm{c}} v$ if and only if $v = xw = wx$ for some $x \in \Gamma^*$.
9. *Power order:* $w \leq_{\mathrm{f}} v$ if and only if $v = w^n$ for some $n \geq 1$.
10. *Equality order:* $w =_{\mathrm{e}} v$ if and only if $w = v$.

Each of these relations is reflexive and all are transitive except the outfix relation. They are ordered by inclusion as follows [3]:

$$=_{\mathrm{e}} \subsetneq \leq_{\mathrm{f}} \subsetneq \leq_{\mathrm{c}} \subsetneq \leq_{\mathrm{d}} \subsetneq \left\{ \begin{matrix} \leq_{\mathrm{p}} \\ \leq_{\mathrm{s}} \end{matrix} \right\} \subsetneq \left\{ \begin{matrix} \leq_{\mathrm{i}} \\ \leq_{\mathrm{o}} \end{matrix} \right\} \subsetneq \leq_{\mathrm{e}} \subsetneq \leq_{\mathrm{u}}.$$

We also consider the infinite chain

$$\omega_1 \subsetneq \omega_2 \subsetneq \cdots \subsetneq w_n \subsetneq \leq_{\mathrm{e}}$$

of binary relations such that $\omega_1 = \leq_{\mathrm{i}}$ and $\leq_{\mathrm{i}} \cup \leq_{\mathrm{o}} \subsetneq \omega_n$ for $n > 1$ which is defined as follows:

Definition 1. *Let $n \in \mathbb{N}$. For $u, v \in \Gamma^*$, let $(u, v) \in \omega_n$ if and only if*

$$\exists u_1, u_2, \ldots, u_n, v_0, v_1, \ldots, v_n \, (u = u_1 u_2 \cdots u_n \wedge v = v_0 u_1 v_1 u_2 \cdots u_n v_n).$$

Note that $\lim_{n\to\infty} \omega_n = \bigcup_{n=1}^{\infty} \omega_n = \leq_{\mathrm{e}}$. That is, the transitive closure of any ω_n is $\leq_{\mathrm{e}}$.

We also use the following property of relations with respect to homomorphisms:

Definition 2. *Let ϱ be a binary relation on Γ^* and let α be a homomorphism from X^* to Y^*. The relation ϱ is* compatible *with α if, for all $x, y \in X^*$, the inclusion $(x, y) \in \varrho$ implies $(\alpha(x), \alpha(y)) \in \varrho$.*

All of the relations listed in Example 1, except $\leq_{\mathrm{u}}$ are compatible with any homomorphism of X^* to Y^*.

We define the following property of relations which is useful for characterizing weak-coding homomorphisms preserving density:

Definition 3. *For a relation ϱ on Γ^*, we say that ϱ is* alphabet preserving *if $(x, y) \in \varrho$ implies $\mathrm{alph}(x) \subseteq \mathrm{alph}(y)$.*

All of the relations listed in Example 1, except $\leq_{\mathrm{u}}$ are alphabet preserving.

4 Densities

Next, we give some definitions from [3] used to describe different types of densities.

Let S be an arbitrary, fixed, non-empty set. Most of the results in this paper concern the special case where S is a free monoid; however, we give the definition in the same generality as [3].

Definition 4. *Let ϱ be a binary relation on S and let $L \subseteq S$. The set L is said to be ϱ-dense if, for every $x \in S$, there is a $y \in L$ such that $(x, y) \in \varrho$.*

For $S = X^*$ and $\varrho = \leq_{\mathrm{i}}$, we arrive at the usual notion of density. Next, we define the property which is studied extensively in the sequel.

Definition 5. *Let ϱ be a relation on Γ^* and let α be a homomorphism of X^* into Y^*. Then α is said to* preserve ϱ-density *if, for any $L \subseteq X^*$, $\alpha(L)$ is ϱ-dense over Y^* whenever L is ϱ-dense over X^*.*

We would also like to be able to compare the families of homomorphisms which preserve different types of density. Naturally, each homomorphism α from X^* into Y^* can be represented by a set of ordered pairs, $(w, \alpha(w))$ for each $w \in X^*$.

Definition 6. *Let ϱ be a binary relation on Γ^*. Then we denote the family of homomorphisms preserving ϱ-density by $H(\varrho)$. We denote the family of weak-coding homomorphisms preserving ϱ-density by $W(\varrho)$. We denote the family of endomorphisms preserving ϱ-density by $E(\varrho)$.*

It follows from Corollary 6.9 of [3] that an endomorphism preserves ϱ-density, for any $\varrho \in \{=_{\mathrm{e}}, \leq_{\mathrm{f}}, \leq_{\mathrm{c}}, \leq_{\mathrm{d}}, \leq_{\mathrm{p}}, \leq_{\mathrm{s}}, \leq_{\mathrm{i}}\}$ if and only if $\varrho|_X$ is a permutation of X. Hence, we can immediately establish the following collapsed hierarchy:

Theorem 1. *[3]* $E(=_{\mathrm{e}}) = E(\leq_{\mathrm{f}}) = E(\leq_{\mathrm{c}}) = E(\leq_{\mathrm{d}}) = E(\leq_{\mathrm{p}}) = E(\leq_{\mathrm{s}}) = E(\leq_{\mathrm{i}})$.

We will show in this paper that this collapsing of the hierarchy will not hold true for arbitrary homomorphisms.

In addition, we use the following definition.

Definition 7. *Let $\alpha : X^* \to Y^*$ be a homomorphism. Then we define* $\mathrm{im}|_X(\alpha) = \{\alpha(a) \mid a \in X\}$ *and* $\max(\alpha) = \max\{|\alpha(a)| \mid a \in X\}$. *In addition, for each $b \in Y$, let $\mu_\alpha(b)$ be the smallest member of $\mathbb{N}_0$ such that $b^{\mu_\alpha(b)} \in \mathrm{im}|_X(\alpha)$ and let $\mu_\alpha(Y) = \max\{\mu_\alpha(b) \mid b \in Y\}$.*

5 Homomorphisms preserving density

We start by including some results from [3] which we use throughout the paper. Again, we will only provide results when S is a free monoid.

Lemma 1. *[3]*

1. *Let $L_1 \subseteq L_2 \subseteq S$ and let ϱ be a binary relation on S. If L_1 is ϱ-dense then L_2 is ϱ-dense.*
2. *Let ϱ_1 and ϱ_2 be two binary relations on S such that $\varrho_1 \subseteq \varrho_2$ and let $L \subseteq S$. If L is ϱ_1-dense then it is ϱ_2-dense.*

Proposition 1. *[3] Let $\alpha : X^* \to Y^*$ be a homomorphism and let ϱ be a binary relation on Γ^* and contained in ω_n for some $n \in \mathbb{N}$. If $\alpha(X^*)$ is ϱ-dense then the following statements hold true:*

1. *For every $a \in Y$, there is an element $b \in X$ and a positive integer $k_{a,b}$ such that $\alpha(b) = a^{k_{a,b}}$.*
2. $|Y| \leq |X|$.

We can rephrase condition (1) above by stating that for every $a \in Y$, it follows that $\mu_\alpha(a) > 0$. Condition (2) implies that for most of the standard binary relations, homomorphisms will only preserve that type of density if the target alphabet is no larger then the domain alphabet.

Next, we show that, for coding or weak coding homomorphisms, preserving density essentially amounts to examining alphabets.

Lemma 2. *Let ϱ be an alphabet preserving binary relation on Γ^* such that there exists some ϱ-dense language over X^* and also let $\alpha : X^* \to Y^*$ be a homomorphism which preserves ϱ-density. Then* $\mathrm{alph}(\alpha(X^*)) = Y$ *and* $\mathrm{alph}(\alpha(X)) = Y$.

Proof. Let L be ϱ-dense over X^*. Then X^* must be ϱ-dense by Lemma 1(1). Thus, for every $u \in X^*$, there exists $v \in X^*$ such that $(u, v) \in \varrho$ and $\mathrm{alph}(u) \subseteq \mathrm{alph}(v)$. Therefore, for every $u' \in Y^*$, there exists $v' \in \alpha(X^*)$ such that $(u', v') \in \varrho$. In particular, for every $y \in Y$, there exists $v' \in \alpha(X^*)$ such that $(y, v') \in \varrho$. But $\mathrm{alph}(y) \subseteq \mathrm{alph}(v')$. Hence, $y \in \mathrm{alph}(\alpha(X^*))$ for every $y \in Y$. □

Lemma 3. *Let ϱ be a binary relation on Γ^* and assume that there exists some ϱ-dense language over X^* and let $\alpha : X^* \to Y^*$ be a weak coding homomorphism. If* $\mathrm{alph}(\alpha(X^*)) = Y$ *and ϱ is compatible with α, then α preserves ϱ-density.*

Proof. Let L be a ϱ-dense language over X^* and let $y = a_1 a_2 \cdots a_n \in Y^*$ with $a_i \in Y, 1 \leq i \leq n$ (the case where $y = \lambda$ is similar). Since $\mathrm{alph}(\alpha(X^*)) = Y$ and α is a weak coding homomorphism, there must exist $b_1, \ldots, b_n \in X$ where $\alpha(b_1 \cdots b_n) = a_1 \cdots a_n$. However, since L is ϱ-dense, there must exist $v \in L$ such that $(b_1 \cdots b_n, v) \in \varrho$. But ϱ is compatible with α, so $(\alpha(b_1 \cdots b_n), \alpha(v)) = (a_1 \cdots a_n, \alpha(v)) \in \varrho$ and $\alpha(v) \in \alpha(L)$. Hence, $\alpha(L)$ is ϱ-dense. □

We sum up the two previous lemmas as follows:

Proposition 2. *Let ϱ be an alphabet preserving binary relation on Γ^* such that there exists some ϱ-dense language over X^*. Also, let $\alpha : X^* \to Y^*$ be a weak coding homomorphism (or indeed a coding homomorphism) whereby ϱ is compatible with α. Then α preserves ϱ-density if and only if* $\mathrm{alph}(\alpha(X^*)) = Y$ *and this holds true if and only if* $\mathrm{alph}(\alpha(X)) = Y$.

Since every reflexive relation will allow X^* to be ϱ-dense, we get the following:

Corollary 1. *Let ϱ be an alphabet preserving, reflexive binary relation on Γ^*. Also, let $\alpha : X^* \to Y^*$ be a weak coding homomorphism (or indeed a coding homomorphism) whereby ϱ is compatible with α. Then α preserves ϱ-density if and only if* $\mathrm{alph}(\alpha(X^*)) = Y$ *and this holds true if and only if* $\mathrm{alph}(\alpha(X)) = Y$.

The conditions of compatibility and alphabet preserving relations as above apply to a large variety of specific relations including the classical notion of density.

Corollary 2. *For $\varrho \in \{=_{\mathrm{e}}, \leq_{\mathrm{f}}, \leq_{\mathrm{c}}, \leq_{\mathrm{d}}, \leq_{\mathrm{p}}, \leq_{\mathrm{s}}, \leq_{\mathrm{i}}, \leq_{\mathrm{o}}, \leq_{\mathrm{e}}\}$, a weak coding homomorphism (or indeed a coding homomorphism) $\alpha : X^* \to Y^*$ preserves ϱ-density if and only if* $\mathrm{alph}(\alpha(X^*)) = Y$ *and this holds true if and only if* $\mathrm{alph}(\alpha(X)) = Y$.

Consequently, the hierarchy for weak-coding homomorphisms completely collapses.

Corollary 3. $W(=_{\mathrm{e}}) = W(\leq_{\mathrm{f}}) = W(\leq_{\mathrm{c}}) = W(\leq_{\mathrm{d}}) = W(\leq_{\mathrm{p}}) = W(\leq_{\mathrm{s}}) = W(\leq_{\mathrm{i}}) = W(\leq_{\mathrm{o}}) = W(\leq_{\mathrm{e}})$.

The following is essentially an extension of results in [3] from endomorphisms to arbitrary homomorphisms. It says that under certain conditions, determining whether $\alpha(X^*)$ is dense is equivalent to determining whether α preserves density.

Proposition 3. *Let $\alpha : X^* \to Y^*$ be a homomorphism and let ϱ be a binary relation on Γ^*. Then the following statements are true.*

1. *If ϱ is transitive and compatible with α and if $\alpha(X^*)$ is ϱ-dense over Y^* then, for every $L \subseteq X^*$ which is ϱ-dense over X^*, also $\alpha(L)$ is ϱ-dense over Y^*.*
2. *If there is an $L \subseteq X^*$ such that $\alpha(L)$ is ϱ-dense over Y^*, then $\alpha(X^*)$ is ϱ-dense over Y^*.*

Proof. (1) Let $y \in Y^*$. As $\alpha(X^*)$ is ϱ-dense, there exists $z \in \alpha(X^*)$ with $(y, z) \in \varrho$. Let $z' \in \alpha^{-1}(z)$. As L is ϱ-dense, there exists $x' \in L$ with $(z', x') \in \varrho$. Let $y' = \alpha(x')$. Hence $y' \in \alpha(L)$ and by compatibility, $(z, y') \in \varrho$. Since $(y, z), (z, y') \in \varrho$, then $(y, y') \in \varrho$ by transitivity.

(2) Let $L \subseteq X^*$ be such that $\alpha(L)$ is ϱ-dense over Y^*. Since $\alpha(L) \subseteq \alpha(X^*)$, then Lemma 1(1) implies that $\alpha(X^*)$ is ϱ-dense over Y^*. □

We use this to show that, under certain natural conditions, determining whether a homomorphism preserves density amounts to checking whether a single regular language is dense.

Proposition 4. *Let $\alpha : X^* \to Y^*$ be a homomorphism, let ϱ be a binary relation on Γ^* which is transitive and compatible with α and assume that there exists $L \subseteq X^*$ which is ϱ-dense over X^*. Then the following conditions are equivalent:*

1. *α preserves ϱ-density.*
2. *$\alpha(X^*)$ is ϱ-dense over Y^*.*
3. *$\mathrm{im}|_X(\alpha)^*$ is ϱ-dense over Y^*.*

Proof. (1) $\Rightarrow$ (2) is true by Propositon 3(2).

(2) $\Leftarrow$ (1) is true by Proposition 3(1).

(2) $\Leftrightarrow$ (3) is true because $\mathrm{im}|_X(\alpha) = \alpha(X)$ and so $\mathrm{im}|_X(\alpha)^* = (\alpha(X))^* = \alpha(X^*)$. □

Furthermore, if ϱ is also reflexive, then $(x, x) \in \varrho$ for every $x \in X^*$ and so X^* is ϱ-dense over X^* and we can simplify the proposition above.

Corollary 4. *Let $\alpha : X^* \to Y^*$ be a homomorphism and let ϱ be a binary relation on Γ^* which is transitive, reflexive and compatible with α. Then X^* is ϱ-dense over X^* and also the following conditions are equivalent:*

1. *α preserves ϱ-density.*
2. *$\alpha(X^*)$ is ϱ-dense over Y^*.*

3. $\mathrm{im}|_X(\alpha)^*$ *is* ϱ*-dense over* Y^*.

We would like to be able to study, formally, the families of homomorphisms preserving density.

Definition 8. *Let* ϱ_1, ϱ_2 *be two binary relations on* Γ^*. *If, for every finite language* L, L^* *is* ϱ_1*-dense implies that* L^* *is* ϱ_2*-dense, then we say* ϱ_1 *is* densely smaller than ϱ_2. *If, for every finite language* L, L^* *is* ϱ_1*-dense if and only if* L^* *is* ϱ_2*-dense, then we say that* ϱ_1 *and* ϱ_2 *are* densely equivalent.

This property is the key to studying families of homomorphisms preserving ϱ-density where ϱ is reflexive, transitive and compatible with arbitrary homomorphisms. We will use it to collapse or separate the different families preserving density.

Theorem 2. *Let* ϱ_1, ϱ_2 *be two binary relations on* Γ^* *which are transitive, reflexive and compatible with arbitrary homomorphisms. Then the following are true:*

1. $\varrho_1 \subseteq \varrho_2$ *implies* $H(\varrho_1) \subseteq H(\varrho_2)$.
2. $H(\varrho_1) \subseteq H(\varrho_2)$ *if and only if* ϱ_1 *is densely smaller than* ϱ_2.
3. $H(\varrho_1) = H(\varrho_2)$ *if and only if* ϱ_1 *is densely equivalent to* ϱ_2.

Proof. (1) For the first statement, let $\alpha : X^* \to Y^*$ be a homomorphism which preserves ϱ_1-density. Then $\mathrm{im}|_X(\alpha)^*$ is ϱ_1-dense over Y^*, by Corollary 4. Then $\mathrm{im}|_X(\alpha)^*$ is ϱ_2-dense over Y^* by Lemma 1(2). Hence, α preserves ϱ_2-density, again by Corollary 4.

(2) Assume that $H(\varrho_1) \subseteq H(\varrho_2)$. Let $L = \{w_1, \ldots, w_n\}$ be a finite language and assume L^* is ϱ_1-dense. Let $a_1, \ldots, a_n$ be n distinct symbols of Γ. Consider the homomorphism α defined by mapping a_i to w_i for each i, $1 \leq i \leq n$. Then $\mathrm{im}|_X(\alpha) = L$. Thus, α must preserve ϱ_1-density since $\mathrm{im}|_X(\alpha)^* = L^*$ is ϱ_1-dense, ϱ_1 is compatible with arbitrary homomorphisms by assumption, and by Corollary 4. By the assumption, α must also preserve ϱ_2-density, and thus L^* must be ϱ_2-dense, again by Corollary 4.

Conversely, assume that ϱ_1 is densely smaller than ϱ_2. Let $\alpha : X^* \to Y^*$ be a homomorphism which preserves ϱ_1-density. Then $\mathrm{im}|_X(\alpha)^*$ is ϱ_1-dense by Corollary 4 and is thus ϱ_2-dense since ϱ_1 is densely smaller than ϱ_2. Hence, α preserves ϱ_2-density, again by Corollary 4.

(3) Immediate from (2). □

So, by the first statement of the previous theorem, we can set up a hierarchy among all relations in Example 1 which are transitive, reflexive and compatible with arbitrary homomorphisms as follows:

Corollary 5. *For* $z \in \{\mathrm{p}, \mathrm{s}\}$, $H(=_{\mathrm{e}}) \subseteq H(\leq_{\mathrm{f}}) \subseteq H(\leq_{\mathrm{c}}) \subseteq H(\leq_{\mathrm{d}}) \subseteq H(\leq_z) \subseteq H(\leq_{\mathrm{i}}) \subseteq H(\leq_{\mathrm{e}})$.

Proposition 5. *Let α be a homomorphism from X^* into Y^*. Then α preserves $=_e$-density if and only if $Y \subseteq \mathrm{im}|_X(\alpha)$.*

Proof. Assume that α preserves $=_e$-density. Thus, for every $w \in Y^*$, $w \in \mathrm{im}|_X(\alpha)^*$ by Corollary 4. Suppose $a \in Y$ such that $a \notin \mathrm{im}|_X(\alpha)$. But $(a, x) \in =_e$ implies $x = a \in \mathrm{im}|_X(\alpha)$, a contradiction. Hence $Y \subseteq \mathrm{im}|_X(\alpha)$.

Conversely, assume $Y \subseteq \mathrm{im}|_X(\alpha)$. Let $w \in Y^*$. Then $w \in \mathrm{im}|_X(\alpha)^*$ and $(w, w) \in =_e$, and thus α preserves $=_e$-density. □

Further, for the case of the embedding relation, we get the following simple characterization.

Proposition 6. *Let $\alpha : X^* \to Y^*$ be a homomorphism and let ϱ be a binary relation on Γ^* which is alphabet preserving, transitive and compatible with α such that $\leq_e \subseteq \varrho$. Then the following are equivalent:*

1. *α preserves $\leq_e$-density.*
2. *α preserves ϱ-density.*
3. *$\mathrm{alph}(\mathrm{im}|_X(\alpha)) = Y$.*

Proof. (1) $\Rightarrow$ (2) It must be true that $\mathrm{im}|_X(\alpha)^*$ is $\leq_e$-dense over Y^* and thus $\mathrm{im}|_X(\alpha)^*$ is ϱ-dense over Y^* by Lemma 1(2). Consequently, α preserves ϱ-density by Corollary 4 and the fact that ϱ must be reflexive since $\leq_e$ is and $\leq_e \subseteq \varrho$.

(2) $\Rightarrow$ (3) This is immediate by Lemma 2 and the fact that ϱ must be reflexive and hence there must exist some ϱ-dense language over X^*.

(3) $\Rightarrow$ (1) Assume $\mathrm{alph}(\mathrm{im}|_X(\alpha)^*) = Y$. Let $w \in Y^*$ with $w = a_1 \cdots a_n, a_i \in Y, 1 \leq i \leq n$. For each a_i, there exists $x_i \in \mathrm{im}|_X(\alpha)$ such that $a_i \in \mathrm{alph}(x_i)$. Let $v = x_1 \cdots x_n \in \mathrm{im}|_X(\alpha)^*$. Also, $w \leq_e v$ and so $\mathrm{im}|_X(\alpha)^*$ is $\leq_e$-dense over Y^*. Hence, α preserves $\leq_e$-density by Corollary 4. □

Using the division relation is identical to using both the prefix and suffix relations.

Proposition 7. *Let $\alpha : X^* \to Y^*$ be a homomorphism. Then α preserves $\leq_d$-density if and only if α preserves both $\leq_p$ and $\leq_s$ density. Thus, $H(\leq_d) = (H(\leq_p) \cap H(\leq_s))$.*

Proof. Assume that α preserves $\leq_d$-density. Thus, $\mathrm{im}|_X(\alpha)^*$ is $\leq_d$-dense over Y^* by Corollary 4. However, $\leq_d \subseteq \leq_p$ and $\leq_d \subseteq \leq_s$ and by Lemma 1(2), $\mathrm{im}|_X(\alpha)^*$ is $\leq_p$-dense and also $\leq_s$-dense. Again, using Corollary 4, α-preserves $\leq_p$-density and also $\leq_s$-density.

Assume that α preserves both $\leq_s$-density and $\leq_p$-density. Therefore, $\mathrm{im}|_X(\alpha)^*$ is $\leq_s$-dense and also $\leq_p$-dense by Corollary 4. Let $w \in Y^*$. Then, there exists $u_1 \in \mathrm{im}|_X(\alpha)^*$ and $u_2 \in \mathrm{im}|_X(\alpha)^*$ such that $w \leq_p u_1$ and $w \leq_s u_2$. Hence, there exists $x, y \in Y^*$ such that $u_1 = wx$ and $u_2 = yw$. Moreover, u_1 and u_2 are in $\mathrm{im}|_X(\alpha)^*$ and so $u_1 u_2 \in \mathrm{im}|_X(\alpha)^*$. Indeed, $w \leq_d u_1 u_2$ and so $\mathrm{im}|_X(\alpha)^*$ is $\leq_d$-dense over Y^* and α preserves $\leq_d$-density by Corollary 4. □

We can collapse part of the hierarchy of Corollary 5 using the commutation and division relations as seen by the following proposition.

Proposition 8. *Let $\alpha : X^* \to Y^*$ be a homomorphism. Then α preserves $\leq_c$-density if and only if α preserves $\leq_f$-density. Thus, $H(\leq_f) = H(\leq_c)$.*

Proof. Assume that α preserves $\leq_c$-density. Hence, $\mathrm{im}|_X(\alpha)^*$ is $\leq_c$-dense over Y^*, by Proposition 4. Let $w \in Y^*$. Then, there exists $v \in \mathrm{im}|_X(\alpha)^*$ such that $w \leq_c v$; that is, there exists $x \in Y^*$ such that $v = wx = xw$. If $w = \lambda$, then $\lambda \leq_f \lambda \in \mathrm{im}|_X(\alpha)^*$. If $x = \lambda$, then $w = v$ and $w \leq_f v$. Assume then, that $w \neq \lambda$ and $x \neq \lambda$. It is well-known (see for example Lemma 1.7 of [5]) that for two words r, s with $r \neq \lambda$ and $s \neq \lambda$, if $rs = sr$, then r and s are powers of a common word. Thus, there exists $u \in Y^*$ such that $x = u^n, w = u^m$ and hence $v = u^{n+m}$ with $u \in Y^+$ and $n, m \in \mathbb{N}_0$. Since $v = u^{m+n} \in \mathrm{im}|_X(\alpha)^*$, we obtain $v' = v^m = u^{m(m+n)} = w^{m+n} \in \mathrm{im}|_X(\alpha)^*$. Indeed, $w \leq_f v' \in \mathrm{im}|_X(\alpha)^*$ and so $\mathrm{im}|_X(\alpha)^*$ is $\leq_f$-dense over Y^* and α preserves $\leq_f$-density by Corollary 4.

Assume that α preserves $\leq_f$-density. Thus, $\mathrm{im}|_X(\alpha)^*$ is $\leq_f$-dense over Y^* by Corollary 4. However, $\leq_f \subseteq \leq_c$ and by Lemma 1(2), $\mathrm{im}|_X(\alpha)^*$ is $\leq_c$-dense. Again, by Corollary 4, α preserves $\leq_c$-density. □

This shows that the converse of Theorem 2(1) is not true because $\leq_f \subsetneq \leq_c$. We observe the following with respect to the difference between prefix and suffix density which will become useful in separating some parts of the hierarchy.

Proposition 9. *Let $L \subseteq Y^*$. Then L is $\leq_p$-dense if and only if L^R is $\leq_s$-dense. Equivalently, L is $\leq_s$-dense if and only if L^R is $\leq_p$-dense.*

Proof. Suppose L is $\leq_p$-dense. Let $w \in Y^*$ and consider w^R. Indeed, $w^R \leq_p v$ for some $v \in L$; that is, $v = w^r x, x \in Y^*$. Then $v^R = x^R w \in L^R$. Furthermore, $(w, v^r) \in \leq_s$.

Conversely, suppose L^R is $\leq_s$-dense. Let $w \in Y^*$ and consider w^R. Then $w^R \leq_s v^R$ for some $v \in L$; that is, $v^R = xw^R$ for some $x \in Y^*$. Indeed, $v = wx^R$ and $(w, v) \in \leq_p$. □

We now turn to separating the hierarchy of Corollary 5 between the infix and embedding relations. The following two families can be separated by showing that they are not densely equivalent using the language $\{aba\}$.

Proposition 10. $H(\leq_i) \subsetneq H(\leq_e)$.

Proof. Consider the language $L_1 = \{aba\}$ and let α be a homomorphism that maps a to aba. Then α preserves $\leq_e$-density by Proposition 6.

Suppose that L_1^* is $\leq_i$-dense. Let $w = bb$. Clearly, bb is not an infix of any word in L_1^*.

Since L_1^* is $\leq_e$-dense but L_1^* is not $\leq_i$-dense, it follows from Theorem 2(3) that $H(\leq_i) \subsetneq H(\leq_e)$. □

In the following, we are able to separate the the homomorphisms which preserve both prefix and suffix density with those that preserve each of prefix and suffix individually. Moreover, prefix and suffix are both incomparable.

For the proofs which follow, for $n \in \mathbb{N}_0$, let $\pi(n)$ be 0 if n is even and 1 otherwise.

Proposition 11. *For* $z \in \{\mathrm{p}, \mathrm{s}\}$,

$$(H(\leq_{\mathrm{p}}) \cap H(\leq_{\mathrm{s}})) \subsetneq H(\leq_z) \subsetneq (H(\leq_{\mathrm{p}}) \cup H(\leq_{\mathrm{s}})).$$

Also, $H(\leq_{\mathrm{p}}) \not\subseteq H(\leq_{\mathrm{s}})$ *and* $H(\leq_{\mathrm{s}}) \not\subseteq H(\leq_{\mathrm{p}})$.

Proof. Consider the language $L = \{a^2, b, ab\}$ and let $Y = \{a, b\}$. We want to show that L^* is $\leq_{\mathrm{p}}$-dense over Y^*. Let $w \in Y^*$. If $w \in \{\lambda\} \cup a^* \cup b^*$ then there exists $v \in L^*$ such that $w \leq_{\mathrm{p}} v$. Otherwise,

$$w = a^{n_1} b^{m_1} a^{n_2} b^{m_2} \cdots a^{n_k} b^{m_k},$$

with $n_1, m_k \geq 0, n_2, \ldots, n_k, m_1, \ldots, m_{k-1} > 0$. Consider,

$$v = (a^2)^{\lfloor n_1/2 \rfloor} (ab)^{\pi(n_1)} (b)^{m_1 - \pi(n_1)} \cdots (a^2)^{\lfloor n_k/2 \rfloor} (ab)^{\lfloor n_k \rfloor} (b)^{n_k - \pi(n_k)}.$$

Indeed, $w \leq_{\mathrm{p}} v$. Thus, L^* is $\leq_{\mathrm{p}}$-dense over $\{a, b\}^*$.

We would like to show that L^* is not $\leq_{\mathrm{s}}$-dense over $\{a, b\}^*$. Suppose otherwise. Consider the word $w = ba$. Then there exists $v = u_1 \cdots u_k$ with $ba \leq_{\mathrm{s}} v$ and $k \geq 1$. Since ba ends with the letter a, necessarily $u_k = a^2$, but $a^2 \neq ba$, a contradiction.

By Theorem 2, this shows that the $\leq_{\mathrm{p}}$ relation is not densely smaller than the $\leq_{\mathrm{s}}$ relation and that $H(\leq_{\mathrm{p}}) \not\subseteq H(\leq_{\mathrm{s}})$. Furthermore, consider the language L^R. By Proposition 9, L^R is $\leq_{\mathrm{s}}$-dense but is not $\leq_{\mathrm{p}}$-dense. Therefore, by Theorem 2, this shows that the $\leq_{\mathrm{s}}$ relation is not densely smaller than the $\leq_{\mathrm{p}}$ relation and that $H(\leq_{\mathrm{s}}) \not\subseteq H(\leq_{\mathrm{p}})$. Therefore, $(H(\leq_{\mathrm{p}}) \cap H(\leq_{\mathrm{s}})) \subsetneq H(\leq_{\mathrm{p}})$ and $(H(\leq_{\mathrm{p}}) \cap H(\leq_{\mathrm{s}})) \subsetneq H(\leq_{\mathrm{s}})$. In addition, $H(\leq_{\mathrm{p}}) \subsetneq (H(\leq_{\mathrm{p}}) \cup H(\leq_{\mathrm{s}}))$ and $H(\leq_{\mathrm{s}}) \subsetneq (H(\leq_{\mathrm{p}}) \cup H(\leq_{\mathrm{s}}))$. □

We still need to separate $=_{\mathrm{e}}$-density from $\leq_{\mathrm{f}}$-density.

Proposition 12. $H(=_{\mathrm{e}}) \subsetneq H(\leq_{\mathrm{f}})$

Proof. Let $L = \{a^2, b, ba, ab\}$. and let $Y = \{a, b\}$. First we define a homomorphism α which maps a to a^2, b to b, c to ba and d to ab. Indeed, α does not preserve $=_{\mathrm{e}}$-density by Proposition 5 and since $a \notin \mathrm{im}|_X(\alpha)$.

We now want to show that $L^* = \mathrm{im}|_X(\alpha)^*$ is $\leq_{\mathrm{f}}$-dense and hence α preserves $\leq_{\mathrm{f}}$-density. Let $w \in Y^*$. If $w \in \{\lambda\} \cup a^* \cup b^*$, then there exists $v \in L^*$ with $w \leq_{\mathrm{f}} v$. Otherwise,

$$w = b^{n_1} a^{m_1} b^{n_2} a^{m_2} \cdots b^{n_k} a^{m_k},$$

with $k \geq 1, n_1, m_k \geq 0, m_1, \ldots, m_{k-1}, n_2, \ldots, n_k > 0$.

Case 1: Assume that m_k is even. Then we rewrite

$$\begin{aligned} w \;=\; & (b)^{n_1} (a^2)^{\lfloor m_1/2 \rfloor} (ab)^{\pi(m_1)} (b)^{n_2 - \pi(m_2)} \cdots \\ & \cdots (a^2)^{\lfloor m_{k-1}/2 \rfloor} (ab)^{\pi(m_{k-1})} (b)^{n_k - \pi(m_{k-1})} (a^2)^{m_k/2}. \end{aligned}$$

Furthermore, $(w, w) \in \leq_{\mathrm{f}}$.

Case 2: Assume that $n_1 > 0$ and m_k is odd. Then we rewrite

$$\begin{aligned} w &= (b)^{n_1-\pi(m_1)}(ba)^{\pi(m_1)}(a^2)^{\lfloor m_1/2 \rfloor} \dots \\ &\quad \dots (b)^{n_k-\pi(m_k)}(ba)^{\pi(m_k)}(a^2)^{\lfloor m_k/2 \rfloor}. \end{aligned}$$

Furthermore, $(w, w) \in \leq_{\mathrm{f}}$.

Case 3: Assume that $n_1 = 0$, m_k is odd and m_1 is even. Then we rewrite

$$\begin{aligned} w &= (a^2)^{m_1/2}(b)^{n_2-\pi(m_2)}(ba)^{\pi(m_2)}(a^2)^{\lfloor m_2/2 \rfloor} \dots \\ &\quad \dots (b)^{n_k-\pi(m_k)}(ba)^{\pi(m_k)}(a^2)^{\lfloor m_k/2 \rfloor}. \end{aligned}$$

Furthermore, $(w, w) \in \leq_{\mathrm{f}}$.

Case 4: Assume that $n_1 = 0$, m_k is odd and m_1 is odd. Then

$$w^2 = a^{m_1} b^{n_2} a^{m_2} \dots b^{n_k} a^{m_k} a^{m_1} b^{n_2} a^{m_2} \dots b^{n_k} a^{m_k}.$$

Then we rewrite

$$\begin{aligned} w^2 &= (a^2)^{\lfloor m_1/2 \rfloor}(ab)^{\pi(m_1)}(b)^{n_2-\pi(m_1)} \dots \\ &\quad \dots (a^2)^{\lfloor m_{k-1}/2 \rfloor}(ab)^{\pi(m_{k-1})}(b)^{n_k-\pi(m_{k-1})} \\ &\quad (a^2)^{(m_k+m_1)/2}(b)^{n_2-\pi(m_2)}(ba)^{\pi(m_2)}(a^2)^{\lfloor m_2/2 \rfloor} \dots \\ &\quad \dots (b)^{n_k-\pi(m_k)}(ba)^{\pi(m_k)}(a^2)^{\lfloor m_k/2 \rfloor}. \end{aligned}$$

Furthermore, $(w, w^2) \in \leq_{\mathrm{f}}$.

Therefore, $L^* = \mathrm{im}|_X(\alpha)^*$ is $\leq_{\mathrm{f}}$-dense and hence α preserves $\leq_{\mathrm{f}}$-density by Corollary 4.

□

Indeed, the last case in the above proof is necessary as shown by the example where $w = aba$. If $w \in L^*$, then $w = u_1 u_2$, where u_1 is necessarily ab and u_2 is a. However, $a \notin L$. That being said, $w^2 = abaaba = (ab)(aa)(ba) \in L^*$.

Note that in the proof above, it would have been immediate to show that L^* was $\leq_{\mathrm{d}}$-dense since $L = L^R$ and thus L^* is both $\leq_{\mathrm{p}}$ and $\leq_{\mathrm{s}}$ dense by Proposition 9 and thus is $\leq_{\mathrm{d}}$-dense by Proposition 11. That being said, it was not immediate that L^* was $\leq_{\mathrm{f}}$-dense.

Next, we separate the union of the homomorphisms preserving prefix and suffix density with those preserving infix density.

Proposition 13. $H(\leq_{\mathrm{p}}) \cup H(\leq_{\mathrm{s}}) \subsetneq H(\leq_{\mathrm{i}})$.

Proof. The inclusion is immediate from Corollary 5. For the strictness, consider the language $L = \{a^2, b, bab, aba, aaab, baaa\} \subseteq Y^*$ where $Y = \{a, b\}$. We first prove the following claim:

Claim 1. *For each* $n > 1, m \geq 0, z = (ba)^m ba^n \in L^*$.

Proof. First assume that n is odd and $m = 0 \mod 3$. Then

$$z = ((bab)(aba))^{m/3}(baaa)(a^2)^{\lfloor n/2 \rfloor - 1}.$$

Assume that n is odd and $m = 1 \mod 3$. Then

$$z = (b)(aba)((bab)(aba))^{(m-1)/3}(a^2)^{\lfloor n/2 \rfloor}.$$

Assume that n is odd and $m = 2 \mod 3$. Then

$$z = ((bab)(aba))^{(m+1)/3}(a^2)^{\lfloor n/2 \rfloor}.$$

Assume that n is even and $m = 0 \mod 3$. Then

$$z = ((bab)(aba))^{m/3}(b)(a^2)^{n/2}.$$

Assume that n is even and $m = 1 \mod 3$. Then

$$z = (bab)((aba)(bab))^{(m-1)/3}(a^2)^{n/2}.$$

Assume that n is even and $m = 2 \mod 3$. Then

$$z = (b)(aba)((bab)(aba))^{(m-2)/3}(b)(a^2)^{n/2}.$$

□

Let $Y_\$ = Y \cup \{\$_1, \$_2, \$_3, \$_4\}$, where $\$_1, \$_2, \$_3, \$_4$ are new symbols. We will show that L^* is $\leq_{\mathrm{i}}$-dense over Y^*. We define four rewriting rules as follows:

$w_1 \to_1 w_2$ if and only if $w_1 = x(ba)^m ba^n cy, w_2 = x\$_1 cy, x, y \in Y_\$^*$, $m > 0, n > 1, c \in \{b, \$_1\}, x, y \in Y_\$^*, ba \not\leq_{\mathrm{s}} x$.

$w_1 \to_2 w_2$ if and only if $w_1 = xba^n cy, w_2 = x\$_2 cy, x, y \in Y_\$^*$, $n > 1, c \in \{b, \$_1, \$_2\}, x, y \in Y_\$^*$.

$w_1 \to_3 w_2$ if and only if $w_1 = xabay, w_2 = x\$_3 y, x, y \in Y_\$^*$.

$w_1 \to_4 w_2$ if and only if $w_1 = xbaby, w_2 = x\$_4 y, x, y \in Y_\$^*$.

Furthermore, for each $i \in \{1, 2, 3, 4\}$, let $w_1 \to_i^{(*)} w_2$ if and only if there exists $n \in \mathbb{N}$ and n words $y_1, \ldots, y_n \in Y_\$^*$ such that $w_1 = y_1 \to_i y_2 \to_i \cdots \to_i y_n = w_2$ and there does not exist any $z \in Y_\$^*$ such that $y_n \to_i z$.

Let $w \in Y^*$. We would like to create $x, y \in Y^*$ such that $xwy \in L^*$. Let w_1, w_2, w_3, w' be any words in $Y_\$^*$ such that $w \to_1^{(*)} w_1 \to_2^{(*)} w_2 \to_3^{(*)} w_3 \to_4^{(*)} w'$. Let $w' = x_1 \$_{\gamma_1} x_2 \$_{\gamma_2} \cdots \$_{\gamma_{k-1}} x_k$ and $w_3 = z_1 \$_{\tau_1} z_2 \$_{\tau_2} \cdots \$_{\tau_{l-1}} z_l$ where $k, l \geq 1, x_j, z_j \in Y^*$ and all \$ symbols are in $\{\$_1, \$_2, \$_3, \$_4\}$.

Now, examining the rewriting rules, we see $(ba)^m ba^n, ba^n \in L^*, m > 0, n > 1$ by Claim 1 and also $aba, bab \in L^*$. Thus, it is sufficient to find $x, y \in Y^*$ such that $xx_1, x_2, x_3, \ldots, x_{k-1}, x_k y \in L^*$ as this implies $xwy \in L^*$.

We will show that for each $i, 2 \leq i \leq k-1$ with $k \geq 2$, it must be true that $a \notin \text{alph}(x_i)$. Suppose otherwise.

First, $a^n, n > 1$ cannot be an infix of any $z_2, \ldots, z_l$ if $l > 1$. Otherwise, $\$_j a^n$ must be an infix of w_3 for some $j \in \{1,2\}$ and the first two rewriting rules can only leave a b or a \$ symbol after a \$ symbol. In addition, $ba^n, n > 1$ cannot be an infix of z_1. Thus, $a^n, n > 1$ cannot be an infix of x_i, otherwise $\$_{\gamma_{i-1}} a^n$ must be an infix of w', γ_{i-1} can be neither 1 nor 2 and if γ_{i-1} is 3 or 4, then $ba^n \leq_{\text{i}} w_3$, a contradiction.

Thus, $x_i \in \{a, ab^n a, ab^m, b^m a \mid n > 1, m \geq 1\}$ since a^n, aba, bab are not infixes of x_i for $n > 1$. Suppose that $x_i = av, v \in Y^*$. Then $\gamma_{i-1} \in \{3,4\}$. If it is 3, then $abaav \leq_{\text{i}} w_2$, a contradiction. If it is 4, then $babav \leq_{\text{i}} w_3$, a contradiction. Hence $x_i = b^m a, m \geq 1$ and $\$_{\gamma_{i-1}} b^m a \$_{\gamma_i} \leq_{\text{i}} w'$. If $\gamma_i = 3$, then $baab \leq_{\text{i}} w_3$, a contradiction. If $\gamma_i = 4$, then $aba \leq_{\text{i}} w_4$. So γ_i is either 1 or 2. If it is 2, then $baba^n c \leq_{\text{i}} w_1, c \in \{b, \$_1\}$. Furthermore, it cannot be 1 since $ba \leq_{\text{s}} b^m a$.

Hence, for each i, $2 \leq i \leq k-1$ with $k \geq 2$, it is true that $x_i \in b^* \subseteq L^*$. Thus, we still need to verify that there exists x, y with $xx_1, x_k y \in L^*$. Indeed, x_1 must be of the form $a^{n_1} b^{n_2} a^{n_3}, n_1, n_2 \geq 0$ and n_3 either 0 or 1. If $n_3 = 0$, then x can be empty if n is even and a if n_1 is odd. Also, $n_2 > 0$ necessarily. So assume $n_2 > 0$ and $n_3 = 1$. We reach a contradiction similarly to the case above. Similarly for the case of x_1, x_k must also equal $a^{n_1} b^{n_2} a^{n_3}, n_1, n_2 \geq 0$, n_3 either 0 or 1. If $n_1 = 0$ or $n_2 = 0$, we are done. Otherwise, $\$_{\gamma_{k-1}}$ must be $\$_3$ or $\$_4$. If it is $\$_3$, then $abaa^{n_1} b \leq_{\text{i}} w_2$, a contradiction. If it is $\$_4$, then $baba \leq_{\text{i}} w_3$, a contradiction. Lastly, if $k = 1$ then $x \in a^* b^* a^*$, and we are done.

To show that L^* is not $\leq_{\text{p}}$-dense, let $w = abba$. It is clear that there does not exist any $v \in L^*$ such that $w \leq_{\text{p}} v$. Thus L^* is not $\leq_{\text{p}}$-dense. Moreover, L^* is not $\leq_{\text{s}}$-dense, since $L = L^R$ and by Proposition 9. Then $H(\leq_{\text{p}}) \cup H(\leq_{\text{s}}) \subsetneq H(\leq_{\text{i}})$. □

Finally, we determine that the inclusion between $H(\leq_{\text{f}})$ and $H(\leq_{\text{d}})$ is strict.

Proposition 14. $H(\leq_{\text{f}}) \subsetneq (H(\leq_{\text{p}}) \cap H(\leq_{\text{s}}))$.

Proof. The inclusion is immediate from Theorem 2. For the strictness of the inclusion, consider the language $L = Y^4 \cup \{b\} \setminus \{baab\}$ over $Y = \{a, b\}$. We need to prove that L^* is both prefix and suffix dense. It is enough to show that it is prefix dense, since $L = L^R$ using Proposition 9.

For $w = a_1 \cdots a_m \in Y^+, m \geq 1, a_j \in Y, 1 \leq j \leq m$, let $\chi(w, n) = d_1 d_2 d_3 d_4, d_i \in Y, 1 \leq i \leq 4$, and $d_i = a_{i+n-1}$ for $1 \leq i \leq 4$, where we define $a_{m+1} = a_{m+2} = a_{m+3} = a$

Let $w \in Y^*$. If $w = \lambda$ then we can construct $v \in L^*$ such that $w \leq_{\text{p}} v$. Assume then that $w \in Y^+$. Consider the two sequences $\{c_i\}_{i \in \mathbb{N}}$ over $\mathbb{N}$ and $\{u_i\}_{i \in \mathbb{N}}$ over Y^* defined as follows:

$$c_i = \begin{cases} 1, & \text{if } i = 1, \\ c_{i-1} + 4, & \text{if } i > 1, c_{i-1} + 4 \leq |w| \text{ and } \chi(w, c_{i-1}) \neq baab, \\ c_{i-1} + 1, & \text{if } i > 1, c_{i-1} + 1 \leq |w| \text{ and } \chi(w, c_{i-1}) = baab, \\ \text{undefined}, & \text{otherwise}, \end{cases}$$

and

$$u_i = \begin{cases} \chi(w, c_i), & \text{if } c_i > 0 \text{ and } \chi(w, c_i) \neq baab, \\ b, & \text{if } c_i > 0 \text{ and } \chi(w, c_i) = baab, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

For each i, $1 \leq i < l$, $|u_1 \cdots u_i| = c_{i+1} - 1$. Let l be the largest integer such that u_l is defined (which must exist by the definitions) and consider the word $v = u_1 \cdots u_l$. Thus l is also the largest integer such that c_l is defined.

Claim 2. *For each i, $1 \leq i \leq l$, $u_1 \cdots u_l \leq_{\mathrm{p}} waaa$.*

Proof. The claim follows when $i = 1$. Let j satisfy $1 \leq j < l$ and assume that $u_1 \cdots u_j \leq_{\mathrm{p}} waaa$. As noted above, $c_{j+1} = |u_1 \cdots u_j| + 1$. Consider u_{j+1} which is the same as $\chi(w, c_{j+1})$ if $\chi(w, c_{j+1}) \neq baab$ and b otherwise. If $j + 1 < l$, then $u_1 \cdots u_{j+1} \leq_{\mathrm{p}} w$ and if $j + 1 = l$, then $u_1 \cdots u_{j+1} \leq_{\mathrm{p}} waaa$. □

Thus, $u_1 \cdots u_l \leq_{\mathrm{p}} waaa$. We would like to still show that $w \leq_{\mathrm{p}} u_1 \cdots u_l$. This follows since $0 \leq |u_1 \cdots u_l| - |w| \leq 3$. Furthermore, $\chi(w, c_i) \neq baab$ is in L and b is also in L. Hence, L^* is $\leq_{\mathrm{s}}$-dense.

We now show that L^* is not $\leq_{\mathrm{f}}$-dense. Assume otherwise and let $w = baab$. Then there exists $n \in \mathbb{N}$ such that $w^n \in L^*$. That is, $(baab)^n = u_1 \cdots u_l, u_i \in L, 1 \leq i \leq l$. Necessarily, $u_1 = b$ and $u_i = aabb$ for each i, $1 \leq i \leq l$, a contradiction. □

Combining Corollary 5 with Propositions 7, 8, 10, 11, 12, 13 and 14, we get the following hierarchy which is far more detailed than the one of Corollary 5.

Theorem 3. *For $z \in \{\mathrm{p}, \mathrm{s}\}$,*

$$H(=_{\mathrm{e}}) \subsetneq H(\leq_{\mathrm{f}}) = H(\leq_{\mathrm{c}}) \subsetneq H(\leq_{\mathrm{d}}) = (H(\leq_{\mathrm{p}}) \cap H(\leq_{\mathrm{s}})) \subsetneq H(\leq_z) \subsetneq (H(\leq_{\mathrm{p}}) \cup H(\leq_{\mathrm{s}})) \subsetneq H(\leq_{\mathrm{i}}) \subsetneq H(\leq_{\mathrm{e}}).$$

Moreover, $H(\leq_{\mathrm{p}}) \not\subseteq H(\leq_{\mathrm{s}})$ and $H(\leq_{\mathrm{s}}) \not\subseteq H(\leq_{\mathrm{p}})$.

This is quite different from the special case for endomorphisms in Theorem 1 and for weak-coding homorphisms in Corollary 3 where the hierarchy collapses.

The property of being densely equivalent was important to establish which parts of the hierarchy collapsed and which did not. Despite this, we used ad hoc techniques in order to determine which two relations were densely equivalent. It is an open question as to whether the results of this hierarchy can be condensed into a more general and concise formulation.

6 Deciding if a homomorphism preserves density

We turn briefly to the question of deciding whether or not a homomorphism preserves different types of density. It turns out that we have already done most of the difficult work for most types. The proposition uses the construct of an a-transducer, which is essentially a nondeterminsitic gsm which can output on λ-input [1].

Proposition 15. *Let $\alpha : X^* \to Y^*$ be an effectively given homomorphism and let ϱ be a binary relation on Γ^* which is transitive, reflexive and compatible with α. Then the following conditions hold:*

1. *If it is decidable whether a regular language $L \subseteq Y^*$ is ϱ-dense over Y^*, then it is decidable whether α preserves density.*
2. *If it is decidable whether $\varrho^{-1}(L) = Y^*$ for every regular language $L \subseteq Y^*$, then it is decidable whether α preserves density.*
3. *If there is an a-transducer M_ϱ which satisfies $M_\varrho(L) = \varrho^{-1}(L)$ for every $L \subseteq Y^*$, then it is decidable whether α preserves density.*

Proof. (1) This follows from Corollary 4 and the fact that $\mathrm{im}|_X(\alpha)$ is finite.

(2) Let $L = \mathrm{im}|_X(\alpha)^*$ which is regular. We can decide whether $\varrho^{-1}(L) = Y^*$. Furthermore, $\varrho^{-1}(L) = Y^*$ if and only if for every $u \in Y^*$, there exists $v \in L$ such that $(u, v) \in \varrho$. Thus, $\mathrm{im}|_X(\alpha)^*$ is ϱ-dense if and only if $\varrho^{-1}(\mathrm{im}|_X(\alpha)^*) = Y^*$, which is decidable.

(3) If there is an a-transducer (or indeed a nondeterministic gsm mapping) M_ϱ which satisfies $M_\varrho(R) = \varrho^{-1}(R)$, for every $R \subseteq Y^*$, then $L' = M_\varrho(\mathrm{im}|_X(\alpha)^*)$ is regular since the family of regular languages is closed under arbitrary a-transductions. Further, the universe problem (given a language L, is $L = Y^*$?) is decidable for the family of regular languages and thus we can decide if $L' = Y^*$. □

As an immediate consequence, we obtain decidability for the five relations $\leq_{\mathrm{e}}, \leq_{\mathrm{p}}, \leq_{\mathrm{s}}, \leq_{\mathrm{i}}, =_{\mathrm{e}}$. For the case of the equality relation, one can decide this property trivially using a much simpler characterization of Proposition 5, whereby, one need only check whether $Y \subseteq \mathrm{im}|_X(\alpha)$ in order to determine whether or not a homomorphism α preserves $=_{\mathrm{e}}$-density. Similarly, for the embedding relation, it follows from Proposition 6 that we need only verify that $\mathrm{alph}(\mathrm{im}|_X(\alpha)) = Y$.

In addition, by Proposition 7, we know a homomorphism preserves $\leq_{\mathrm{d}}$-density if and only if it preserves both prefix and suffix density. Hence, by Proposition 15, we can decide whether a homomorphism preserves $\leq_{\mathrm{d}}$-density.

The problem is not so easy to decide for power and commutation density however. We need to start with the following characterization.

Proposition 16. *Let $\alpha : X^* \to Y^*$ be a homomorphism. Then the following are equivalent:*

1. *α preserves $\leq_{\mathrm{f}}$-density.*
2. *α preserves $\leq_{\mathrm{c}}$-density.*
3. *For every $w \in Y^*$, there exists $v \in \mathrm{im}|_X(\alpha)^*$ and an integer n, $1 \leq n \leq \max(\alpha)$ such that $v = w^n$.*

4. *For every* $w \in Y^*$, *there exists an integer* n, $1 \leq n \leq \max(\alpha)$ *and either* $w \in \inf(\mathrm{im}|_X(\alpha))$ *and there exists* $v \in \mathrm{im}|_X(\alpha)^*$ *such that* $v = w^n$ *or there exists* $n+1$ *ordered pairs,*

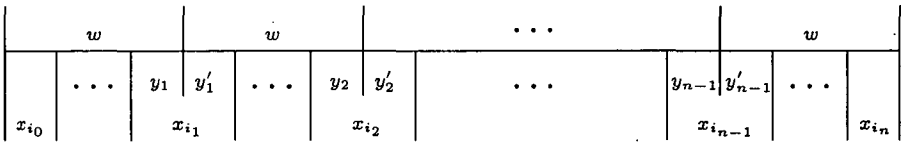
$$(y_0, y_0'), (y_1, y_1'), \ldots, (y_n, y_n'),$$

with $y_i y_i' \in \mathrm{im}|_X(\alpha)$ *for* $0 \leq i \leq n$, $y_0 = y_n' = \lambda, \{y_0', \ldots, y_{n-1}', y_1, \ldots, y_n\} \subseteq Y^+$ *and* $(y_i')^{-1} w (y_{i+1})^{-1} \in \mathrm{im}|_X(\alpha)^*$, $0 \leq i < n$.

Proof. (1) $\Leftrightarrow$ (2) Immediate from Lemma 8.

(1) $\Rightarrow$ (3) Assume that α preserves $\leq_{\mathrm{f}}$-density. Let $w \in Y^*$. Then there exists a minimal integer $n \geq 1$ and $v \in \mathrm{im}|_X(\alpha)^*$ with $v = w^n$. If $n \leq \max(\alpha)$ then we are done. Assume that $n > \max(\alpha)$. Thus, $w^n = x_1 x_2 \cdots x_m, x_i \in \mathrm{im}|_X(\alpha)$. Thus, for some integer j, $1 \leq j \leq \max(\alpha)$, there exists k_1, k_2 with $k_1 < k_2$ such that $|x_1 \cdots x_{k_1}| = l_1|w| + j$, $l_1 \in \mathbb{N}_0$ and $|x_1 \cdots x_{k_2}| = l_2|w| + j, l_2 \in \mathbb{N}_0$. Thus, consider $v' = x_1 \cdots x_{k_1} x_{k_2+1} \cdots x_m$. Then, $v' = w^{n+l_1-l_2} \in \mathrm{im}|_X(\alpha)^*$. This contradicts the minimality of n.

(3) $\Rightarrow$ (4) Let $w \in Y^*$. Then there exists $v \in \mathrm{im}|_X(\alpha)^*$ and an integer n, $1 \leq n \leq \max(\alpha)$ such that $v = w^n$ with n minimal. If $w \in \inf(\mathrm{im}|_X(\alpha))$, then we are done. Assume that $w \notin \inf(\mathrm{im}|_X(\alpha))$. Thus, $v = w^n = x_1 x_2 \cdots x_m$, $1 \leq n \leq \max(\alpha), x_i \in \mathrm{im}|_X(\alpha), 1 \leq i \leq m$ with $m > 1$. If $n = 1$, then there exists $(\lambda, x_1), (x_m, \lambda)$ such that $x_1, x_m \in \mathrm{im}|_X(\alpha)$ and $(x_1)^{-1} x_1 \cdots x_m (x_m)^{-1} \in \mathrm{im}|_X(\alpha)^*$ (m must be greater than 1 and if $m = 2$ then $(x_1)^{-1} x_1 \cdots x_m (x_m)^{-1} = \lambda \in \mathrm{im}|_X(\alpha)^*$). Assume that $n > 1$. Since $w \notin \inf(\mathrm{im}|_X(\alpha))$, there exists $i_1, i_1, \ldots, i_n$ such that $i_0 = 1 < i_1 < \cdots < i_{n-1} < i_n = m$ where $|x_1 \cdots x_{i_{j-1}}| < j|w| < |x_1 \cdots x_{i_j}|$, $1 \leq j \leq n-1$.



For each j, consider the ordered pair (y_j, y_j') where y_j is the prefix of x_{i_j} of length $j|w| - |x_1 \cdots x_{i_{j-1}}|$ and $y_j' = (y_j)^{-1} x_{i_j}$. Both $y_j \neq \lambda$ and $y_j' \neq \lambda$ by the minimality of n. So we have ordered pairs,

$$(\lambda, x_1), (y_1, y_1'), \ldots, (y_{n-1}, y_{n-1}'), (x_m, \lambda).$$

Also, let $y_0' = x_1$ and $y_n = x_m$. Indeed, $y_j y_j' \in \mathrm{im}|_X(\alpha)$ for all j, $1 \leq j \leq n-1$ and $x_1, x_m \in \mathrm{im}|_X(\alpha)$. Moreover, for each j, $1 \leq j \leq n-2, x_{i_j+1} \cdots x_{i_{j+1}-1} \in \mathrm{im}|_X(\alpha)^*, x_2 \cdots x_{i_1-1} \in \mathrm{im}|_X(\alpha)^*$ and $x_{i_{n-1}+1} \cdots x_{m-1} \in \mathrm{im}|_X(\alpha)^*$. Hence, for each k, $0 \leq k < n$, $(y_k')^{-1} w (y_{k+1})^{-1} \in \mathrm{im}|_X(\alpha)^*$.

(4) $\Rightarrow$ (1) Let $w \in Y^*$. Then there exists $1 \leq n \leq \max(\alpha)$ satisfying the stated assumptions. If $w \in \inf(\mathrm{im}|_X(\alpha))$ then there exists $v \in \mathrm{im}|_X(\alpha)^*$ with $v = w^n$, by assumption. Otherwise, there exists $n+1$ ordered pairs, $(y_0, y_0'), \ldots, (y_n, y_n')$ where $y_i y_i' \in \mathrm{im}|_X(\alpha), 0 \leq i \leq n, y_0 = y_n' = \lambda$ and $\{y_0', \ldots, y_{n-1}', y_1, \ldots, y_n\} \subseteq Y^+$ and $z_i = (y_i')^{-1} w (y_{i+1})^{-1} \in \mathrm{im}|_X(\alpha)^*, 0 \leq i < n$. Consider

$$v = y_0' z_0 y_1 y_1' z_1 y_2 \cdots y_{n-1} y_{n-1}' z_{n-1} y_n.$$

Indeed, $v \in \mathrm{im}|_X(\alpha)^*$. Furthermore, $v = w^n$. □

We can then use this characterization to show that determining whether a homomorphism preserves power density amounts to deciding the universe problem on regular languages. The proof uses an NFA which nondeterministically guesses the $n+1$ ordered pairs in the proof above.

Proposition 17. *Let $\alpha : X^* \to Y^*$ be an effectively given homomorphism. Then we can construct a regular language L whereby $L = Y^*$ if and only if α preserves $\leq_{\mathrm{f}}$-density.*

Proof. Let M be a nondeterministic finite automata which nondeterministically guesses an integer n, $1 \leq n \leq \max(\alpha)$ and $n+1$ ordered pairs

$$(y_0, y_0'), \ldots, (y_n, y_n'),$$

where $y_0 = y_n' = \lambda$, $y_i y_i' \in \mathrm{im}|_X(\alpha)$ for each i, $0 \leq i \leq n$ and also the set $\{y_1', \ldots, y_n', y_2, \ldots, y_{n+1}\} \subseteq Y^+$. Then, on input $w \in Y^* \setminus \inf(\mathrm{im}|_X(\alpha))$ (intersect Y^* with the complement of $\inf(\mathrm{im}|_X(\alpha))$ which is regular), in parallel, for each i, $0 \leq i \leq n$, M verifies that $(y_i')^{-1} w (y_{i+1})^{-1} \in \mathrm{im}|_X(\alpha)^*$. Let $L_1 = L(M)$. Furthermore, let $L_2 = \{w \mid w \in \inf(\mathrm{im}|_X(\alpha)), w^n \in \mathrm{im}|_X(\alpha)^*, 1 \leq n \leq \max(\alpha)\}$. It is clear that L_2 is finite and can be effectively constructed. Let $L = L_1 \cup L_2$. Then L is regular and $L = Y^*$ if and only if α preserves $\leq_{\mathrm{f}}$-density, by Proposition 16. □

Combining Proposition 17 and 16, and the fact that the universe problem for regular languages is decidable [2], we get decidability for $\leq_{\mathrm{c}}$- and $\leq_{\mathrm{f}}$-density. Collecting the decidability over all relations together, we obtain:

Corollary 6. *Let $\alpha : X^* \to Y^*$ be an effectively given homomorphism. Then it is decidable whether α preserves ϱ-density where $\varrho \in \{=_{\mathrm{e}}, \leq_{\mathrm{f}}, \leq_{\mathrm{c}}, \leq_{\mathrm{d}}, \leq_{\mathrm{p}}, \leq_{\mathrm{s}}, \leq_{\mathrm{i}}, \leq_{\mathrm{e}}\}$.*

References

[1] Ginsburg, S. *Algebraic and Automata-Theoretic Properties of Formal Languages.* North-Holland Publishing Company, Amsterdam, 1975.

[2] Hopcroft, J. E. and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[3] Jürgensen, H., Kari, L., and Thierrin, G. Morphisms preserving densities. *International Journal of Computer Mathematics*, 78:165–189, 2001.

[4] Jürgensen, H. and Konstantinidis, S. Codes. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, volume 1, pages 511–607. Springer-Verlag, Berlin, 1997.

[5] Shyr, H.J. *Free Monoids and Languages.* Hon Min Book Company, Taichung, third edition, 2001.

Received 23rd October 2008

Acta Cybernetica 19 (2009) 517–536.

Complexity of Problems Concerning Reset Words for Some Partial Cases of Automata

Pavel Martyugin*

Abstract

A word w is called a reset word for a deterministic finite automaton $\mathscr{A}$ if it maps all states of $\mathscr{A}$ to one state. A word w is called a compressing to M states for a deterministic finite automaton $\mathscr{A}$ if it maps all states of $\mathscr{A}$ to at most M states. We consider several subclasses of automata: aperiodic, $\mathscr{D}$-trivial, monotonic, partially monotonic automata and automata with a zero state. For these subclasses we study the computational complexity of the following problems. Does there exist a reset word for a given automaton? Does there exist a reset word of given length for a given automaton? What is the length of the shortest reset word for a given automaton? Moreover, we consider complexity of the same problems for compressing words.

Keywords: synchronization, automata, reset words, computational complexity

1 Synchronization

A *deterministic finite automaton* (DFA) $\mathscr{A}$ is a triple $\langle Q, \Sigma, \delta\rangle$, where Q is a finite set of *states*, Σ is a finite *alphabet*, and $\delta : Q \times \Sigma \to Q$ is a totally defined *transition function*. The function δ extends in a unique way to an action $Q \times \Sigma^* \to Q$ of the free monoid Σ^* over Σ; this extension is also denoted by δ. We denote $\delta(q, w)$ by $q.w$. We also define for $S \subseteq Q$, $w \in \Sigma^*$, $\delta(S, w) = S.w = \{q.w | q \in S\}$.

A DFA $\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter at which state in Q it started: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$. Any word w with this property is said to be a *reset* or *synchronizing word* for the automaton.

In [1], Černý produced for each n a synchronizing automaton with n states and 2 input letters whose shortest reset word has length $(n-1)^2$ and conjectured that these automata represent the worst possible case, that is, every synchronizing automaton with n states can be reset by a word of length $(n-1)^2$. The conjecture is arguably the most longstanding open problem in the combinatorial theory of finite automata.

*Ural State University, E-mail: martugin@mail.ru

Upper bounds within the confines of the Černý conjecture have been obtained for the maximum length of the shortest reset words for synchronizing automata in some special classes, see, e.g., [2, 6, 7, 9, 10, 13, 14]. Some of these classes are considered in this paper.

One of these classes is the class of commutative DFA. An automaton $\mathscr{A} = (Q, \Sigma, \delta)$ is said to be *commutative* if its transformation monoid is commutative, that is, for every state $q \in Q$ and for all letters $a, b \in \Sigma$, $\delta(q, ab) = \delta(q, ba)$. Rystsov in [10] proved that every commutative synchronizing automaton with n states has a reset word of length $n - 1$. This means that the Černý conjecture is true for commutative automata.

Another class of DFA considered by Rystsov in [11] is a class of automata with simple idempotents. Let $\mathscr{A} = (Q, \Sigma, \delta)$ be a DFA. Let $a \in \Sigma$. If $\delta(Q, a) = Q$, then the letter a is called a *permutation*. If $\delta(Q, a^2) = \delta(Q, a)$, then the letter a is called an *idempotent*. If the letter a is an idempotent and $|\delta(Q, a)| = |Q| - 1$ then a is called a *simple idempotent*. If for DFA $\mathscr{A}$ all letters are permutations or simple idempotents, then such an automaton is called an *automaton with simple idempotents*. Every n-state synchronizing automaton with simple idempotents admits a reset word of length $2(n-1)^2$ (see [11]).

Another natural class of DFA is a class of automata with a zero state. A state z of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is said to be a *zero* state if $\delta(z, a) = z$ for all $a \in \Sigma$. It is clear that a synchronizing automaton may have at most one zero state and each word that resets a synchronizing automaton possessing a zero state must bring all states to the zero state. A rather straightforward argument shows that any n-state synchronizing automaton can be reset by a word of length $\frac{n(n-1)}{2}$, see, e.g., [10]. This upper bound is in fact tight because, for each n, there exists a synchronizing automaton with n states and $n-1$ input letters which cannot be reset by any word of length less than $\frac{n(n-1)}{2}$. In [5] it was proved that for each integer $n \geq 8$, there exists a synchronizing automaton with n states and 2 input letters such that the length of the shortest reset word for this automaton is $\left\lceil \frac{n^2+6n-16}{4} \right\rceil$.

Another two classes of DFA can be defined via Greens Relations $\mathscr{H}$ and $\mathscr{D}$. Let M be a transition monoid of some DFA $\mathscr{A}$. Let $U, V \subseteq M$. Denote $UV = \{uv | u \in U, v \in V\}$. The relations $\mathscr{H}$ and $\mathscr{D}$ can be defined on any monoid. Let M be a finite monoid and $u, v \in M$ then $u \mathscr{H} v \Leftrightarrow uM = vM$ and $Mu = Mv$; $u \mathscr{D} v \Leftrightarrow MuM = MvM$. An automaton is called *aperiodic* or *$\mathscr{H}$-trivial* if its transition semigroup has no nontrivial $\mathscr{H}$-classes. An automaton is called *$\mathscr{D}$-trivial* if its transition semigroup has no nontrivial $\mathscr{D}$-classes. Every synchronizing strongly connected aperiodic automaton has a reset word of length $\lfloor \frac{n(n+1)}{6} \rfloor$ (see [9]). Moreover, any synchronizing aperiodic automaton has a reset word of length $\frac{n(n-1)}{2}$.

We also consider another three classes of automata. A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ is called *monotonic* if its state set admits a linear order $\leq$ such that for each letter $a \in \Sigma$ the transformation $\delta(\_, a)$ of Q preserves $\leq$ in the sense that $\delta(q, a) \leq \delta(q', a)$ whenever $q \leq q'$. Every synchronizing monotonic automaton has a reset word of length at most $n - 1$ (see [6]). A DFA is called *cyclically monotonic* if its state set

admits a cyclic order preserving by an action of any letter. Every synchronizing cyclically monotonic automaton has a reset word of length at most $(n-1)^2$ (see [2]).

A deterministic incomplete automaton is an automaton with a partial transition function (in an incomplete automaton the value $\delta(q,a)$ can be undefined on some pairs (q,a)). A deterministic incomplete automaton is called *partially monotonic* if its state set admits a linear order $\leq$ such that for each $a \in \Sigma$ the partial transformation $\delta(\_,a)$ preserves the restriction of order to the domain of the transformation. Every incomplete automaton $\mathscr{A} = (Q,\Sigma,\delta)$ can be transformed to a complete automaton $\mathscr{A}' = (Q \cup \{end\},\Sigma,\delta')$ by an adding of one state *end* such that if for some $q \in Q, a \in \Sigma$ the value $\delta(q,a)$ is undefined then $\delta'(q,a) = end$. If the automaton $\mathscr{A}$ is partially monotonic then we call the DFA $\mathscr{A}'$ *partially monotonic* too. Every partially monotonic DFA is an aperiodic automaton with a zero state. Every synchronizing partially monotonic automaton has a reset word of length at most $n-1$ $(n-1)+\lfloor\frac{n-2}{2}\rfloor$. On the other hand, for any $n \geq 6$ there exists a 2-letter partially monotonic DFA such that its shortest reset word has length $(n-1)+\lfloor\frac{n-2}{2}\rfloor$ (see [8]).

2 Complexity

It is natural to consider computational complexity of various problems arising from the study of automata synchronization. Most natural questions are: is the given automaton synchronizing or not, and what is the length of the shortest reset word for a given automaton?

In [2], Eppstein presented an algorithm which checks whether the given DFA $\mathscr{A} = (Q,\Sigma,\delta)$ is synchronizing. This algorithm works within $O(|\Sigma| \cdot |Q|^2) + |Q|^3$ times bound. Moreover, for a synchronizing automaton this algorithm finds some reset word. This word can be not a shortest reset word for $\mathscr{A}$.

In [3], Salomaa proved that the following problem is NP-hard. Let a DFA $\mathscr{A}$ and an integer number L be given. The question is the following: is there a word of length $\leq L$ resetting the automaton $\mathscr{A}$. This problem remains NP-complete even if the input automaton has a 2-letter alphabet.

In [4], Samotij considered another problem. Let a DFA $\mathscr{A}$ and an integer number L be given. The question is the following: is the length of the shortest reset word for automaton $\mathscr{A}$ equal to L. It turns out that this problem is NP-hard and co-NP-hard. To prove these statements the construction from [3] can be applied. This method gives also that the considered problem remains NP-hard and co-NP-hard for 2-letter automata.

There is a further natural problem for DFA. Given a DFA $\mathscr{A} = (Q,\Sigma,\delta)$, we define a rank of the word $w \in \Sigma^*$ as the cardinality of the image of the transformation $\delta(\_,w)$ of the set Q. (Thus, in this terminology reset words are exactly the words of rank 1). In 1978 Pin conjectured that for every M, if an n-state automaton admits a word of rank at most M, then it also has a word with rank at most M and of length $(n-M)^2$. But Kari [12] has found a counterexample in

the case $n - M = 4$. Let some word have a rank M in the automaton $\mathscr{A}$. Then we say that this word *compresses* the automaton $\mathscr{A}$ to M states. We consider the complexity of the problem of finding the length of the shortest word compressing a given automaton to M states.

In the present paper we consider these problems for partial cases of the DFA. We give a denotation to any considered class of DFA. Let

- DFA be the denotation of the class of all DFA,
- COM be the denotation of the class of commutative automata,
- SIMPID be the denotation of the class of automata with simple idempotents,
- APER be the denotation of the class of aperiodic automata,
- $\mathscr{D}$-TRIV be the denotation of the class of $\mathscr{D}$-trivial automata,
- MON be the denotation of the class of monotonic automata,
- PMON be the denotation of the class of partially monotonic automata,
- ZERO be the denotation of the class of automata with a zero state.

Let C be some class of DFA. Let us give formal definitions of the following problems.

Problem: $SYN(C)$
Input: A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ from the class C.
Question: Is there a reset word $w \in \Sigma^*$ for the automaton $\mathscr{A}$?

Problem: $SYN(C, \leq L)$
Input: A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ from the class C and an integer $L > 0$.
Question: Is there a reset word $w \in \Sigma^*$ of length $\leq L$ for the automaton $\mathscr{A}$?

Problem: $SYN(C, = L)$
Input: A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ from the class C and an integer $L > 0$.
Question: Does the shortest reset word $w \in \Sigma^*$ for the automaton $\mathscr{A}$ have length L?

Problem: $COMP(C, M, \leq L)$
Input: A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ from the class C and integers $M, L > 0$.
Question: Is there a word $w \in \Sigma^*$ of length $\leq L$ such that $|\delta(Q, w)| \leq M$?

Problem: $COMP(C, M, = L)$
Input: A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ from the class C and integers $M, L > 0$.
Question: Does the shortest word $w \in \Sigma^*$ such that $|\delta(Q, w)| \leq M$ have length L?

In applications automata usually have not an arbitrary alphabet, but an alphabet of fixed size. If the input of some PROBLEM contains only automata having an alphabet of size $\leq k$ for some fixed k, then we call such a problem k-PROBLEM (for example, k-$SYN(ZERO)$).

Let the DFA $\mathscr{A} = (Q, \Sigma, \delta)$, $|Q| = n, |\Sigma| = k$ and the integer $L > 0$ be input data. In this paper we obtain the following results.

- The problems $SYN(ZERO)$, $SYN(\mathscr{D}\text{-}TRIV)$, $SYN(MON)$, $SYN(PMON)$ can be solved in time $O(nk)$.
- The problems $SYN(ZERO, \leq L)$, $SYN(APER, \leq L)$, $SYN(\mathscr{D}\text{-}TRIV, \leq L)$, $SYN(PMON, \leq L)$ are NP-complete together with the corresponding k-problems for $k \geq 2$.
- The problems $SYN(ZERO, = L)$, $SYN(APER, = L)$, $SYN(\mathscr{D}\text{-}TRIV, = L)$, $SYN(PMON, = L)$ are NP-hard and co-NP-hard together with the corresponding k-problems for $k \geq 2$.
- The problem $SYN(COM)$ can be solved in time $O(kn \ln n)$.
- The problem $SYN(COM, \leq L)$ is NP-complete.
- The problem k-$SYN(COM, \leq L)$ for some fixed $k \geq 1$ can be solved in time $O(n^k \ln n)$.
- The problem $SYN(COM, = L)$ is NP-hard and co-NP-hard.
- The problem k-$SYN(COM, = L)$ for some fixed $k \geq 1$ can be solved in time $O(n^k \ln n)$.
- The problem $SYN(SIMPID, \leq L)$ is NP-complete.
- The problem 2-$SYN(SIMPID, \leq L)$ can be solved in time $O(n)$.
- The problem $SYN(SIMPID, = L)$ is NP-hard and co-NP-hard.
- The problem 2-$SYN(SIMPID, = L)$ can be solved in time $O(n)$.
- The problems $COMP(MON, M, \leq L)$ and k-$COMP(MON, M, \leq L)$ for $k \geq 2$ are NP-complete.
- The problems $COMP(MON, M, = L)$ and k-$COMP(MON, M, = L)$ for $k \geq 2$ are NP-hard and co-NP-hard.

The problems $SYN(APER)$ and $SYN(SIMPID)$ can be solved in time $O(kn^2)$ because these problems can be solved in such time for arbitrary input DFA (not necessarily aperiodic or automata with simple idempotents), see [2]. It follows from [2] that the problems $SYN(MON, \leq L)$ and $SYN(MON, = L)$

can be solved in time $O(kn^2)$ (the algorithm from [2] works with cyclically monotonic automata; any monotonic automaton is cyclically monotonic). The only open question is what is the complexity of the problems $k\text{-}SYN(SIMPID, \leq L)$ and $k\text{-}SYN(SIMPID, = L)$ for fixed $k > 2$?

For the sequel, we need some notation. We denote by $|Q|$ the cardinality of a set Q. We denote the set of all subsets of a set Q by 2^Q. For a word $w \in \{a, b\}^*$, we denote by $|w|$ the length of w and by $w[i]$, where $1 \leq i \leq |w|$, the i^{th} letter in w from the left. If $1 \leq i \leq j \leq |w|$, we denote by $w[i, j]$ the word $w[i] \cdots w[j]$.

3 Checking the synchronizability

Proposition 3.1. *Let C be a subclass of DFA, then*

1. The problems $SYN(C)$ and $k\text{-}SYN(C)$ for $k \geq 1$ can be solved in time $O(n^2k)$, where n is a number of states, k is an alphabet size.

2. The problems $SYN(C, \leq L)$ and $k\text{-}SYN(C, \leq L)$ for $k \geq 1$ belong to NP.

Proof. From [2] we have that the problem $SYN(DFA)$ can be solved in time $O(n^2k)$. From [3] we have that the problem $SYN(DFA, \leq L)$ belongs to NP. The problems $SYN(C)$ and $k\text{-}SYN(C)$ are partial cases of the problem $SYN(DFA)$. Therefore they can be solved in time $O(n^2k)$ too. The problems $SYN(C, \leq L)$ and $k\text{-}SYN(C, \leq L)$ are partial cases of the problem $SYN(DFA, \leq L)$. Therefore these problems belong to NP. □

Proposition 3.2. *The problem $SYN(ZERO)$ can be solved in time $O(nk)$, where n is a number of states, k is an alphabet size.*

Proof. We construct an algorithm checking whether the DFA $\mathscr{A}$ is synchronizing or not. Let q_0 be a zero state in automaton $\mathscr{A}$. Our algorithm is a breadth first search from the state q_0 in the automaton $\mathscr{A}$. We move along the arrows in back direction. If some state $q \in Q$ was not visited during the search, then the automaton is not synchronizing, because there is no word $w \in \Sigma$ such that $q.w = q_0$. Otherwise the automaton is synchronizing. Every arrow can be used no more than once during the search. Therefore, the complexity of the algorithm is $O(nk)$, because the automaton $\mathscr{A}$ contains exactly nk arrows. □

Proposition 3.3. *The problems $SYN(\mathscr{D}\text{-}TRIV)$ and $SYN(PMON)$ can be solved in time $O(nk)$, where n is a number of states, k is an alphabet size.*

Proof. Any $\mathscr{D}$-trivial automaton is $\mathscr{R}$-trivial. For every $\mathscr{R}$-trivial automaton $\mathscr{A} = (Q, \Sigma, \delta)$ the linear order $\leq$ can be defined on the set Q such that for any $q \in Q, a \in \Sigma$, $q.a \geq q$. Let $Q = \{1, \ldots, n\}$ and $1 < 2 < \ldots < n$, then for any word $w \in \Sigma^*$, $n.w = n$. Therefore, the state n is a zero state in the automaton $\mathscr{A}$. From Proposition 3.2, we have that the problem $SYN(\mathscr{D}\text{-}TRIV)$ can be solved in time $O(nk)$.

There is a state *end* in any partially monotonic DFA. The state *end* is always a zero state. Therefore, any partially monotonic automaton has a zero state. Hence, the problem $SYN(PMON)$ can be solved in time $O(nk)$. □

Proposition 3.4. *The problem $SYN(MON)$ can be solved in time $O(nk)$, where n is a number of states, k is an alphabet size.*

Proof. The automaton $\mathscr{A}$ is monotonic, therefore there is a linear order $\leq$ on the set Q such that for any $q_1, q_2 \in Q$ and $a \in \Sigma$ if $q_1 \leq q_2$ then $q_1.a \leq q_2.a$. Let $Q = \{1, \ldots, n\}$ and $1 < 2 < \ldots < n$, then for any word $w \in \Sigma^*$, $1.w \leq 2.w \leq \ldots \leq n.w$. Therefore, a word w is synchronizing if and only if $1.w = n.w$.

Let $p = \max\{q \in Q | \exists w \in \Sigma^*, 1.w = q\}$. Let $v \in \Sigma^*$ such that $1.v = p$. If $n.w > p$ for every word $w \in \Sigma^*$ then $n.w > p \geq 1.w$ by the choice of p, therefore the automaton $\mathscr{A}$ is not synchronizing. If there is a word $u \in \Sigma^*$ such that $n.u \leq p$, then the word uv resets the automaton $\mathscr{A}$ into the state p (because for any $q \in Q$, $q.uv \leq n.uv \leq p.v \leq p$, and $q.uv \geq 1.uv \geq 1.v = p$). Our algorithm finds words u and v. The letter $u[1]$ can be found in time $O(k)$, then the letter $u[2]$ can be found in time $O(k)$ and so on. Therefore, the word u can be found in time $O(nk)$. The word v can be found in the same way in time $O(nk)$. Hence, the problem $SYN(MON)$ can be solved in time $O(nk)$. □

4 Finding the length of the shortest reset words

We will use the classical NP-complete problem SAT to prove the NP-hardness of different problems.

Problem: *SAT*

Input: A set of Boolean variables $x_1, \ldots, x_n$ (the value of any variable can be 0 or 1) and a set of p Boolean expressions (which are called *clauses*) of kind $c_i(x_1, \ldots, x_n) = y_1^i \vee \ldots \vee y_{s_i}^i$, $i \in \{1, \ldots, p\}$ where $y_j^i \in \{x_\ell | \ell \in \{1, \ldots, n\}\} \cup \{\neg x_\ell | \ell \in \{1, \ldots, n\}\}$.

Question: Is there values for variables $x_1, \ldots, x_n \in \{0, 1\}$ such that for any $i \in \{1, \ldots, p\}$, $c_i(x_1, \ldots, x_n) = 1$?

In [3] and [4] the complexity of the problems $SYN(DFA, \leq L)$ and $SYN(DFA, = L)$ were considered. In the next proposition we formulate these results and recall a construction from the proof of these results (the construction is taken from [3]).

Proposition 4.1.

1. The problems $SYN(DFA, \leq L)$ and k-$SYN(DFA, \leq L)$ for $k \geq 2$ are NP-hard.

2. The problems $SYN(DFA, = L)$ and k-$SYN(DFA, = L)$ for $k \geq 2$ are NP-hard and co-NP-hard.

Proof. 1. The problems $SYN(DFA, \leq L)$ and k-$SYN(DFA, \leq L)$ for $k \geq 2$ belong to NP (see Proposition 3.1).

We reduce the problem SAT to the problem 2-$SYN(DFA, \leq L)$. Let the set of clauses $c_1(x_1, \ldots, x_n), \ldots, c_p(x_1, \ldots, x_n)$ over the variables $x_1, \ldots, x_n$ be an input

of the problem SAT. We are going to construct a 2-letter automaton $\mathscr{A}_{dfa} = (Q, \{a,b\}, \delta)$ and a number L. Let $\Sigma = \{a,b\}$, $Q = \{q(m,i) | i \in \{1,\dots,p\}, m \in \{1,n+1\} \cup \{end\}$. Let $i \in \{1,\dots,p\}, m \in \{1,\dots,n\}$, then

$$q(m,i).a = \begin{cases} end, & \text{if } x_m \text{ is contained in } c_i \text{ without } \neg \\ q(m+1,i), & \text{otherwise} \end{cases},$$

$$q(m,i).b = \begin{cases} end, & \text{if } \neg x_m \text{ is contained in } c_i \\ q(m+1,i), & \text{otherwise} \end{cases}.$$

For $i \in \{1,\dots,p\}$, let $q(n+1,i).a = q(n+1,i).b = end$, $end.a = end.b = end$. We put $L = n$.

An example of the automaton $\mathscr{A}_{dfa}$ is represented by the Figure 1. The action of the letter a is denoted by solid lines. The action of the letter b is denoted by dotted lines. The figure contains three columns of states. In the i-th column there are states of kind $q(m,i)$ for fixed i. In any horizontal row there are states of kind $q(m,i)$ for some fixed m.

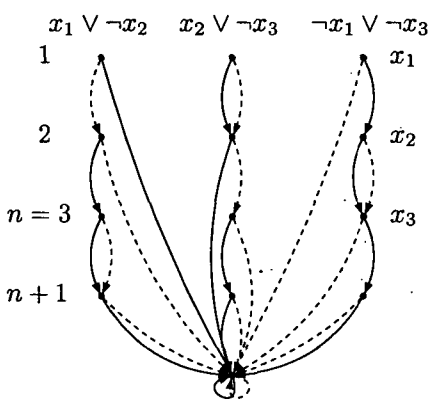


Figure 1: Automaton $\mathscr{A}_{dfa}$ for clauses $x_1 \vee \neg x_2$, $x_2 \vee \neg x_3$, $\neg x_1 \vee \neg x_3$

It is easy to see that the automaton $\mathscr{A}_{dfa}$ has polynomial size. It follows from [3] that there exists a reset word of length L for the automaton $\mathscr{A}_{dfa}$ if and only if there exist values of the variables $x_1,\dots,x_p$ such that $c_1(x_1,\dots,x_n) = \dots = c_p(x_1,\dots,x_n) = 1$. Therefore, the problem 2-$SYN(DFA, \leq L)$ is NP-complete.

Let $k > 2$. The problem 2-$SYN(DFA, \leq L)$ reduces to the problem k-$SYN(DFA, \leq L)$ with adding $k-2$ new letters. Any new letter acts identically on the set of states. Moreover, the problem 2-$SYN(DFA, \leq L)$ is a partial case of the problem $SYN(DFA, \leq L)$. Therefore, the problems $SYN(DFA, \leq L)$ and k-$SYN(DFA, \leq L)$ for $k \geq 2$ are NP-complete.

2. The proof of the NP-hardness of the problems $SYN(DFA, = L)$ and k-$SYN(DFA, = L)$ for $k \geq 2$ is the same. To prove the co-NP-hardness we should construct the automaton $\mathscr{A}_{dfa}$ again using the clauses $c_1,\dots,c_p$. Now we put $L = n+1$. Then we ask: does the shortest reset word for the automaton $\mathscr{A}_{dfa}$

has length L. The length of the shortest reset word for the automaton $\mathscr{A}_{dfa}$ is less or equal to $L = n + 1$, because there are only $n + 1$ rows of states and the state *end* in the automaton $\mathscr{A}_{dfa}$ and every letter maps every state from the some row to a state from the next row or to the state *end*. The shortest reset word for the automaton $\mathscr{A}_{dfa}$ has length L if and only if there are no values for the variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = \ldots = c_p(x_1, \ldots, x_n) = 1$. Therefore the problems $SYN(DFA, = L)$ and $k\text{-}SYN(DFA, = L)$ for $k \geq 2$ are co-NP-hard. □

Now we consider the same problems for some subclasses of automata. It is very easy to prove the NP-hardness for these subclasses because the automaton $\mathscr{A}_{dfa}$ belongs to each of these classes.

Proposition 4.2. *1. The problems $SYN(ZERO, \leq L)$, $SYN(APER, \leq L)$, $SYN(\mathscr{D}\text{-}TRIV, \leq L)$, $SYN(PMON, \leq L)$ are NP-complete together with the corresponding k-problems for $k \geq 2$.*

2. The problems $SYN(ZERO, = L)$, $SYN(APER, = L)$, $SYN(\mathscr{D}\text{-}TRIV, = L)$, $SYN(PMON, = L)$ are NP-hard and co-NP-hard together with the corresponding k-problems for $k \geq 2$.

Proof. Let us prove that the DFA $\mathscr{A}_{dfa} = (Q, \{a, b\}, \delta)$ is a $\mathscr{D}$-trivial, aperiodic, partially monotonic automaton with a zero state. Is can be easily proved that the state *end* is a zero state in the automaton $\mathscr{A}_{dfa}$. Therefore $\mathscr{A}_{dfa}$ is an automaton with a zero state.

Now we prove that the automaton $\mathscr{A}_{dfa}$ is partially monotonic. Let us define the linear order $\leq$ on the set $Q\backslash\{end\}$. We put $q(m_1, i_1) \leq q(m_2, i_2)$, if $i_1 < i_2$, or $i_1 = i_2$ and $m_1 \leq m_2$. It is easily proved that if $q, q' \in Q\backslash\{end\}$, $q \leq q'$ and $\alpha \in \{a, b\}$ such that $q.\alpha \neq end$ and $q'.\alpha \neq end$ then $q.\alpha \leq q'.\alpha$. Therefore $\mathscr{A}_{dfa}$ is partially monotonic.

Now we prove that the automaton $\mathscr{A}_{dfa}$ is $\mathscr{D}$-trivial. Let M be a transition monoid of the DFA $\mathscr{A}_{dfa}$. Let words $u, v \in \Sigma^*$ act on the state Q as different transitions. This means that there is a state $q \in Q$ such that $q.u \neq q.v$. It is easy to see that $q \neq end$, therefore $q = q(m, i)$ for some $m \in \{1, \ldots, n\}, i \in \{1, \ldots, p\}$. We put $q(n+2, i) = end$ for any $i \in \{1, \ldots, p\}$. From the definition of the $\mathscr{A}_{dfa}$, we have that $q.u = q(m_1, i)$ and $q.v = q(m_2, i)$ for some $m_1 \neq m_2$. Let $m_1 < m_2$. Let us take $f, g \in M$. The state $q.fvg$ is equal to $q(m_3, i)$ for some $m_3 \geq m_2 > m_1$. Hence, $fvg \neq \lambda u \lambda$, for any $f, g \in M$, where λ is an empty word. Therefore, $MuM \neq MvM$ and $(u, v) \notin \mathscr{D}$. Thus, the automaton $\mathscr{A}_{dfa}$ is $\mathscr{D}$-trivial. Every $\mathscr{D}$-trivial automaton is aperiodic, therefore $\mathscr{A}_{dfa}$ is aperiodic. □

5 Commutative automata

Proposition 5.1. *The problem $SYN(COM)$ can be solved in time $O(kn \ln n)$, where $n = |Q|$, $k = |\Sigma|$.*

Proof. Let $\Sigma = \{a_1, \ldots, a_k\}$. Every synchronizing commutative DFA $\mathscr{A}$ with n states can be synchronized by a word of length $n - 1$ (see [10]). Whence, there

exists a reset word for the automaton $\mathscr{A}$, containing at most $n-1$ occurrences of the letter a_1, at most $n-1$ occurrences of the letter a_2, and so on. If we add one extra letter to a reset word then we obtain a reset word again. Moreover, the letters contained in a reset word can be permuted and the obtained word will be a reset word again. Therefore, the word $w = a_1^{n-1}a_2^{n-1}\dots a_k^{n-1}$ synchronizes every n-state automaton with alphabet $\{a_1,\dots,a_k\}$. It means that $|Q.w| = 1$ if and only if the automaton $\mathscr{A}$ is synchronizing. The value $Q.w$ can be calculated in time $O(kn\ln n)$, using the famous idea of the fast power calculation.

If the transformation defined by some word u is known, then for every set $S \subseteq Q$, the set $S.u$ can be calculated in time $O(n)$. Let $a \in \Sigma$. The transformation defined by the word a^{n-1} can be calculated in time $O(n\ln n)$ using the fast power calculation. Therefore, if we already know the set $Q.a_1^{n-1}a_2^{n-1}\dots a_{j-1}^{n-1}$, then the calculation of the set $Q.a_1^{n-1}a_2^{n-1}\dots a_{j-1}^{n-1}a_j^{n-1}$ takes time $O(n\ln n)$. Whence, the set $Q.w$ can be calculated in time $O(kn\ln n)$. When all the calculations are finished, we should look at the cardinality of the set $Q.w$. If $|Q.w| = 1$, then the automaton $\mathscr{A}$ is synchronizing, if $|Q.w| \neq 1$, then $\mathscr{A}$ is not synchronizing. The proposition is proved. □

Proposition 5.2. *Let $k \geq 1$, then the problems k-$SYN(COM, \leq L)$ and k-$SYN(COM, = L)$ can be solved in time $O(n^k \ln n)$, where $n = |Q|$.*

Proof. Every synchronizing commutative automaton $\mathscr{A} = (Q, \Sigma, \delta)$ with n states has a reset word of length at most $n-1$. Let $\Sigma = \{a_1,\dots,a_k\}$. If we take a shortest reset word and place its letters in the alphabetic order, we obtain a shortest reset word of kind $a_1^{s_1}a_2^{s_2}\dots a_k^{s_k}$. Therefore we can search a shortest reset word among the words of this kind and length at most $n-1$.

If the numbers $s_1,\dots,s_{k-1}$ are fixed, then the set $Q.a_1^{s_1}a_2^{s_2}\dots a_{k-1}^{s_{k-1}}$ can be found. For fixed numbers $s_1,\dots,s_{k-1}$, the number s_k can be found by the binary search as a minimal s such that $|Q.a_1^{s_1}a_2^{s_2}\dots a_{k-1}^{s_{k-1}}a_k^{s}| = 1$. Every set of the form $Q.a_1^{s_1}a_2^{s_2}\dots a_{k-1}^{s_{k-1}}$ and the sets $Q.a_1^{s_1}a_2^{s_2}\dots a_k^{s}$ (which appear during the binary search) can be calculated in time $O(kn\ln n)$ using the fast power calculation. The number k is fixed, hence $O(kn\ln n) = O(n\ln n)$. Therefore, the shortest reset word of language $a_1^{s_1}a_2^{s_2}\dots a_k^{*}$ can be found in time $O(n\ln n)$ for every vector $(s_1,\dots,s_{k-1})$ with $s_1+\dots+s_{k-1} < n$. The number of these vectors is $\binom{n+k-2}{k-1} = O(n^{k-1})$. To find the answer, the length of the shortest reset word should be compared with L. The complete working time of the algorithm is $O(n^k \ln n)$. The proposition is proved. □

Proposition 5.3. *The problem $SYN(COM, \leq L)$ is NP-complete. The problem $SYN(COM, = L)$ is NP-hard and co-NP-hard.*

Proof. Let us consider the proof of NP-completeness of the problem $SYN(DFA, \leq L)$ from [4] (it is called there SYNCH WORD). We prove that the automaton from this proof is commutative. We reduce the problem SAT to the problem $SYN(COM, \leq L)$. Let the set of clauses $c_1(x_1,\dots,x_n),\dots,c_p(x_1,\dots,x_n)$ over the boolean variables $x_1,\dots,x_n$ be an input

of the problem *SAT*. We are going to construct an automaton $\mathscr{A}_{com} = (Q, \Sigma, \delta)$ and a number L such that there exists a reset word of length L for the automaton $\mathscr{A}_{com}$ if and only if there exist values of the variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = 1, \ldots, c_p(x_1, \ldots, x_n) = 1$.

Let $\mathscr{A}_{com} = (Q, \Sigma, \delta)$, where $Q = \{v_1, \ldots, v_n, q_1, \ldots, q_p, end\}$, $\Sigma = \{a_1, b_1, \ldots, a_n, b_n\}$, and the function δ is the following:

$$\text{For } m \in \{1, \ldots, n\},\ v_m.a_m = v_m.b_m = end,$$

$$\text{for } j \in \{1, \ldots, n\},\ j \neq m,\ v_m.a_j = v_m.b_j = v_m,$$

$$q_i.a_m = \begin{cases} \text{For } i \in \{1, \ldots, p\},\ m \in \{1, \ldots, n\}, \\ end, \quad \text{if } x_m \text{ is contained in } c_i \text{ without } \neg \\ q_i, \quad \text{otherwise} \end{cases},$$

$$q_i.b_m = \begin{cases} end, & \text{if } \neg x_m \text{ is contained in } c_i \\ q_i, & \text{otherwise} \end{cases},$$

$$\text{For } m \in \{1, \ldots, n\},\ end.a_m = end.b_m = end.$$

An example of the automaton $\mathscr{A}_{com}$ for clauses $x_1 \vee \neg x_2$, $x_2 \vee \neg x_3$ and $\neg x_1 \vee \neg x_3$ is represented by Figure 2. We put $L = n$. It is evident that the size of the automaton $\mathscr{A}_{com}$ is polynomial with respect to the input size.

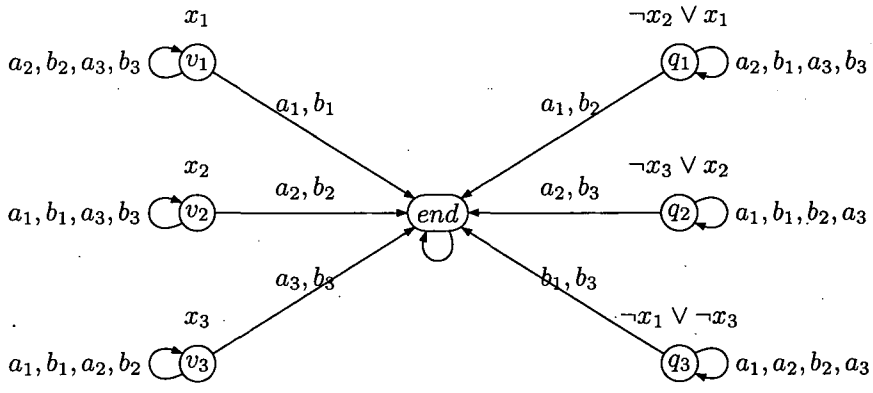


Figure 2: Automaton $\mathscr{A}_{com}$ for clauses $x_1 \vee \neg x_2$, $x_2 \vee \neg x_3$, $\neg x_1 \vee \neg x_3$

There is no letter which maps the state *end* to another state. Whence, the automaton $\mathscr{A}_{com}$ can be synchronized only to the state *end* and the states $v_1, \ldots, v_n$ should be mapped to the state *end*. Therefore, for every $m \in \{1, \ldots, n\}$ one of the letters a_m and b_m should be used. This means that there is no reset word of length less than n.

Let there exist values of the variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = 1, \ldots, c_p(x_1, \ldots, x_n) = 1$. Consider the word w of length n where

$$w[m] = \begin{cases} a_m, & \text{if } x_m = 1 \\ b_m, & \text{if } x_m = 0 \end{cases} \quad \text{for } m \in \{1, \ldots, n\}.$$

For every $i \in \{1, \ldots, p\}$, $c_i(x_1, \ldots, x_n) = 1$. Therefore there exists a number m such that x_m without $\neg$ is contained in c_i and $x_m = 1$, in this case $w[m] = a_m$; or $\neg x_m$ is contained in c_i and $x_m = 0$, in this case $w[m] = b_m$. In both cases $q_i.w[m] = end$. Therefore $Q.w = \{end\}$.

Assume that there exists a reset word w of length n for the automaton $\mathscr{A}_{com}$. For each $m \in \{1, \ldots, n\}$ there exists one and only one of letters a_m and b_m in the word w. We put

$$x_m = \begin{cases} 1, & \text{if } a_m \text{ is contained in } w \\ 0, & \text{if } b_m \text{ is contained in } w \end{cases}$$

If the letter a_m maps the state q_i to the state *end*, then the value $x_m = 1$ provides $c_i = 1$. If the letter b_m maps the state q_i to the state *end*, then the value $x_m = 0$ provides $c_i = 1$. Thus $c_1(x_1, \ldots, x_n) = 1, \ldots, c_p(x_1, \ldots, x_n) = 1$.

Now we prove that the automaton $\mathscr{A}_{com}$ is commutative. Let $\alpha, \beta \in \Sigma$.

- Let $m \in \{1, \ldots, n\}$. If $\alpha \notin \{a_m, b_m\}$ and $\beta \notin \{a_m, b_m\}$, then $v_m.\alpha\beta = v_m.\beta\alpha = v_m$, else $v_m.\alpha\beta = v_m.\beta\alpha = end$.
- Let $i \in \{1, \ldots, p\}$. If $q_i.\alpha \neq end$ and $q_i..\beta \neq end$, then $q_i.\alpha\beta = q_i.\beta\alpha = q_i$, else $q_i.\alpha\beta = q_i.\beta\alpha = end$.
- $end.\alpha\beta = end.\beta\alpha = end$.

Whence the automaton $\mathscr{A}_{com}$ is commutative.

For any synchronizing commutative automaton $\mathscr{A} = (Q, \Sigma, \delta)$ the length of the shortest reset word does not exceed $|Q| - 1$ (see [10]). This means that if $L \geq |Q| - 1$, then for solving problem $SYN(COM, \leq L)$ it is enough to check whether the automaton is synchronizing. It can be done in polynomial time. Thus, the problem $SYN(COM, \leq L)$ is contained in the class NP.

The proof of the NP-hardness of the problem $SYN(COM, = L)$ is the same. Our proof of the co-NP-hardness and the proof from [4] are different (the proof from [4] is more complicated). To prove the co-NP-hardness we should construct the automaton $\mathscr{A}_{com}$ again using the clauses $c_1, \ldots, c_p$. Now we put $L = n+1$. Then we ask the question: is it correct that a shortest reset word for the automaton $\mathscr{A}_{com}$ has length L. The shortest reset word for the automaton $\mathscr{A}_{com}$ has length L if and only if there are no values for the variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = \ldots = c_p(x_1, \ldots, x_n) = 1$. Therefore the problem $SYN(COM, = L)$ is co-NP-hard. The proposition is proved. □

Thus, the problems $k\text{-}SYN(COM, \leq L)$ and $k\text{-}SYN(COM, = L)$ for a fixed number $k \geq 1$ can be solved in polynomial time, but at the same time the problems $SYN(COM, \leq L)$ and $SYN(COM, = L)$ are hard.

6 Automata with simple idempotents

A construction similar to the construction of the automaton $\mathscr{A}_{com}$ can be used to estimate the computational complexity of problems stated for the class of DFA with simple idempotents. As for commutative automata, the complexity of the problems $SYN(SIMPID, \leq L)$ and $SYN(SIMPID, = L)$ differ from the corresponding 2-problems.

Proposition 6.1. *The problem $SYN(SIMPID, \leq L)$ is NP-complete. The problem $SYN(SIMPID, = L)$ is NP-hard and co-NP-hard.*

Proof. We reduce the problem SAT to the problem $SYN(SIMPID, \leq L)$.

Let the input of the problem SAT is a set of clauses $c_1(x_1, \ldots, x_n), \ldots, c_p(x_1, \ldots, x_n)$ over the variables $x_1, \ldots, x_n$. We are going to construct an automaton $\mathscr{A}_{sid} = (Q, \Sigma, \delta)$ and a number L such that there exists a reset word of length L for the automaton $\mathscr{A}_{sid}$ if and only if there exist values of the variables $x_1, \ldots, x_p$ such that $c_1(x_1, \ldots, x_n) = 1, \ldots, c_p(x_1, \ldots, x_n) = 1$.

Let $\mathscr{A}_{sid} = (Q, \Sigma, \delta)$, where $Q = \{v_1, \ldots, v_n, q_1, \ldots, q_p, u, r_1, \ldots, r_p, end\}$, $\Sigma = \{a_1, b_1, \ldots, a_n, b_n, z, y_1, \ldots, y_p\}$, and the function δ is the following:

For $m \in \{1, \ldots, n\}$, $v_m.a_m = v_m.b_m = u$, $u.a_m = u.b_m = v_m$

For $j \in \{1, \ldots, n\}$, $j \neq m$, $v_m.a_j = v_m.b_j = v_m$

For $i \in \{1, \ldots, p\}$, $m \in \{1, \ldots, n\}$,
if x_m is contained in c_i without $\neg$, then $q_i.a_m = r_i$, $r_i.a_m = q_i$
else $q_i.a_m = q_i$, $r_i.a_m = r_i$
if $\neg x_m$ is contained in c_i, then $q_i.b_m = r_i$, $r_i.b_m = q_i$
else $q_i.b_m = q_i$, $r_i.b_m = r_i$

For $q \in Q$, $q.z = \begin{cases} end, & \text{if } q = u \\ q, & \text{otherwise} \end{cases}$

For $i \in \{1, \ldots, p\}$, and $q \in Q$, $q.y_i = \begin{cases} end, & \text{if } q = r_i \\ q, & \text{otherwise} \end{cases}$

For $m \in \{1, \ldots, n\}$, $i \in \{1, \ldots, p\}$, $end.a_m = end.b_m = end.y_i = end.z = end$.

An example of the automaton $\mathscr{A}_{sid}$ for clauses $x_1 \vee \neg x_2$, $x_2 \vee \neg x_3$ and $\neg x_1 \vee \neg x_3$ is represented by Figure 3. We put $L = 2n + 2p + 1$. It is obvious that the size of the automaton $\mathscr{A}_{sid}$ is a function with respect to the input size. The letters $a_1, \ldots, a_n, b_1, \ldots, b_n$ are permutations of the set Q, the letters $z, y_1, \ldots, y_p$ are simple idempotents. Whence $\mathscr{A}_{sid}$ is an automaton with simple idempotents.

The state *end* can be mapped only to the state *end*. Therefore the automaton $\mathscr{A}_{sid}$ can be synchronized only to the state *end*. The automaton $\mathscr{A}_{sid}$ contains $n + 2p + 2$ states. At most one state (except *end*) can be mapped to the state *end* under an action of one letter. The only letters that map some states to the state *end* are $z, y_1, \ldots, y_p$. This means that every reset word should contain at least $n + 2p + 1$ letters from the set $\{z, y_1, \ldots, y_p\}$. Furthermore, a word maps the states

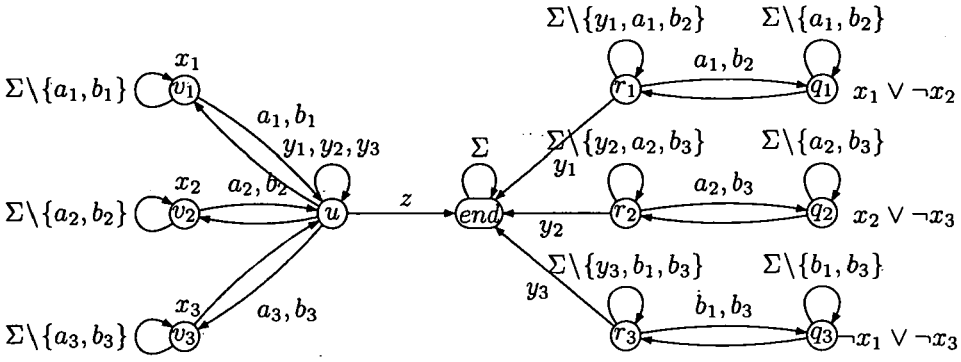


Figure 3: Automaton $\mathscr{A}_{sid}$ for clauses $x_1 \vee \neg x_2$, $x_2 \vee \neg x_3$, $\neg x_1 \vee \neg x_3$

$v_1, \ldots, v_n$ to the state *end* only if it contains a letter from every pair $\{a_m, b_m\}$ for $m \in \{1, \ldots, n\}$. Therefore there is no reset word of length less than $2n + 2p + 1$.

Let there exist values of the variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = \ldots = c_p(x_1, \ldots, x_n) = 1$. We construct a word $w \in \Sigma^*$ of length $2n + 2p + 1$. We put $w[1, p+1] = zy_1 \ldots y_p$. The states $u, r_1, \ldots, r_p$ map to the state *end* under the action of the word $w[1, p+1]$. We put $w[p+2] = \begin{cases} a_1, & \text{if } x_1 = 1 \\ b_1, & \text{if } x_1 = 0 \end{cases}$ and $w[p+3] = z$. In this case the state v_1 also maps to the state *end*. Let after the variable x_1 get value, t_1 of clauses $c_{i_1(1)}, \ldots, c_{i_1(t_1)}$ become true (=1 independently of the values of $x_2, \ldots, x_n$). Then we put $w[p+4, p+t_1+3] = y_{i_1(1)} \ldots y_{i_1(t_1)}$, and the states $q_{i_1(1)}, \ldots, q_{i_1(t_1)}$ map to the state *end*. In the same way, we put $w[p+t_1+4] = \begin{cases} a_2, & \text{if } x_2 = 1 \\ b_2, & \text{if } x_2 = 0 \end{cases}$ and $w[p+t_1+5] = z$. Let then the variable x_2 get value, t_2 of the clauses $c_{i_2(1)}, \ldots, c_{i_2(t_2)}$ become true (we consider only the clauses been false before the variable x_2 get a value). We put $w[p+t_1+6, p+t_1+t_2+5] = y_{i_2(1)} \ldots y_{i_2(t_2)}$. We repeat this process for all variables. For every variable x_i a number t_i is defined. All the clauses becomes true after all the variables get their values. Therefore $t_1 + \ldots + t_n = p$, and the length of the word w is equal to $2n + 2p + 1$. Furthermore, $Q.w = end$.

Assume that there exists a reset word w of length $2n + 2p + 1$ for the automaton $\mathscr{A}_{sid}$. If w is a reset word, then it contains at least $n + 2p + 1$ letters from the set $\{z, y_1, \ldots, y_p\}$ and just one letter from every pair $\{a_m, b_m\}$. The states $q_1, \ldots, q_p$ map to the states $r_1, \ldots, r_p$ under the action of word w, because for each $m \in \{1, \ldots, n\}$ the letter a_m or the letter b_m is contained in w. We put $x_m = \begin{cases} 1, & \text{if } a_m \text{ is contained in } w \\ 0, & \text{if } b_m \text{ is contained in } w \end{cases}$, $m \in \{1, \ldots, n\}$. If the letter a_m maps the state q_i to r_i, then the equality $x_m = 1$ provides $c_i(x_1, \ldots, x_n) = 1$; if the letter b_m maps the state q_i to r_i, then the equality $x_m = 0$ provides $c_i(x_1, \ldots, x_n) = 1$. Thus all the clauses are true.

A length of every reset word for the given synchronizing automaton $\mathscr{A} = (Q, \Sigma, \delta)$ with simple idempotents does no exceed $2(|Q| - 1)^2$ (see [11]). It means that if $L \geq 2(|Q| - 1)^2$, then it is enough to check whether the automaton is synchronizing. It can be done in polynomial time. We can check in polynomial time whether the word $w \in \Sigma^*$ of length L is a reset word for the automaton $\mathscr{A}$. Thus, the problem $SYN(SIMPID, \leq L)$ is in the class NP.

The proof of the NP-hardness of the problem $SYN(SIMPID, = L)$ is the same. To prove the co-NP-hardness we should construct the automaton $\mathscr{A}_{sid}$ again using the clauses $c_1, \ldots, c_p$ and put $L = 2n + 2p + 2$. The shortest reset word for the automaton $\mathscr{A}_{sid}$ has length L if and only if there are no values for the variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = \ldots = c_p(x_1, \ldots, x_n) = 1$. Therefore, $SYN(SIMPID, = L)$ is co-NP-hard. The proposition is proved. □

Proposition 6.2. *The problems 2-$SYN(SIMPID)$, 2-$SYN(SIMPID, \leq L)$ and 2-$SYN(SIMPID, = L)$ can be solved in time $O(n)$, where $n = |Q|$.*

Proof. Let us consider the automaton $\mathscr{A} = (Q, \{a, b\}, \delta)$ and let $|Q| = n$. If a and b are permutations, then for $n > 1$ the automaton $\mathscr{A}$ is not synchronizing. If a and b are simple idempotents, then for $n > 3$ the automaton $\mathscr{A}$ is not synchronizing too. All variants of the automaton for $n \leq 3$ can be easily considered in a constant time. Therefore, we can assume that a is a simple idempotent and b is a permutation.

Let $q_1.a = q_2 \neq q_1$ for $q_1, q_2 \in Q$. The permutation b can be represented as a product of simple cycles. If the permutation b consists of more than two cycles, then the automaton $\mathscr{A}$ is not synchronizing, because the letter a can merge states from at most two cycles. Let b consists of two cycles. In this case the states q_1 and q_2 should contain in different cycles C_1 and C_2. Moreover, if the cycle C_2 consists of more then one state, then the automaton $\mathscr{A}$ is not synchronizing, because different states from the cycle C_2 cannot be merged. In this case the automaton $\mathscr{A}$ looks like the automaton represented by Figure 4.

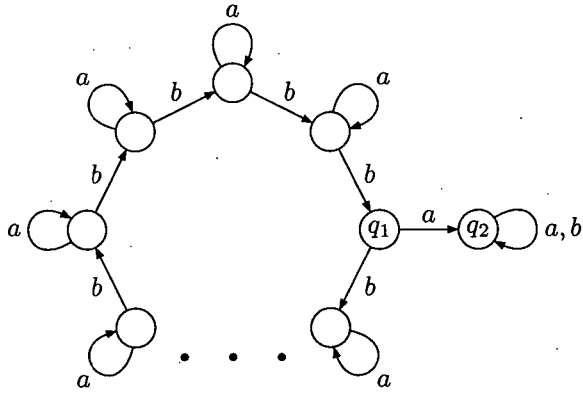


Figure 4: Automaton with two cycles

It is not difficult to check that the word $a(ba)^{n-2}$ is a shortest reset word for $\mathscr{A}$ in this case.

Let the letter b act on the set Q as a single cycle. Let $Q = \{1, \ldots, n\}$, $q_1 = 1$, $q_2 = p$ for some $p \in \{1, \ldots, n\}$, and $n.b = n-1, \ldots, 2.b = 1, 1.b = n$. Such an automaton is represented by Figure 5.

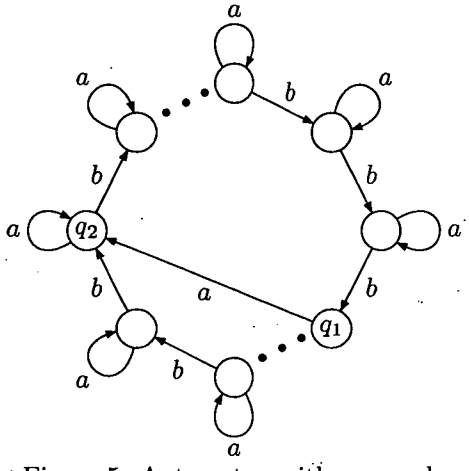


Figure 5: Automaton with one cycle

For $q_2 = n$, the automaton $\mathscr{A}$ is a Černý automaton with n states described in [1]. Is it not difficult to obtain that if $\gcd(n, p) > 1$ then the automaton $\mathscr{A}$ is not synchronizing. If $\gcd(n, p) = 1$, then the word $a(b^{p-1}a)^{n-2}$ is a shortest reset word for the automaton $\mathscr{A}$. The proof of this fact is very similar to the proof from [1] (in [1] it was proved that the word $a(b^{n-1}a)^{n-2}$ is a shortest reset word for the Černý automaton). We skip this proof here.

Let an automaton $\mathscr{A}$ be given. There exists a very simple algorithm taking time $O(n)$ and checking whether the automaton $\mathscr{A}$ can be represented by Figure 4 or by Figure 5. This algorithm finds the states q_1 and q_2, calculates the length of cycles of permutation b. If the states q_1 and q_2 are contained in one cycle, then the algorithm also finds the distance between q_1 and q_2 along the cycle. Finally, the algorithm compares one of the values $a(ba)^{n-2}$ (for the case of two cycles) and $a(b^{p-1}a)^{n-2}$ (for the case of one cycle) with the number L. The proposition is proved.

□

Thus, the problems 2-$SYN(SIMPID, \leq L)$ and 2-$SYN(SIMPID, = L)$ can be solved in polynomial time, but at the same time the problems $SYN(SIMPID, \leq L)$ and $SYN(SIMPID, = L)$ are hard. The question about the computational complexity of the problems k-$SYN(SIMPID, \leq L)$ and k-$SYN(SIMPID, = L)$ for $k > 2$ is open.

7 Finding the length of shortest compressing words

It was proved in [2] that the shortest synchronizing word for a given k-letter cyclically monotonic automaton with n states can be found in time $O(n^2k)$. Every monotonic automaton is cyclically monotonic too. Hence the problems $SYN(MON, \leq L)$, $k\text{-}SYN(MON, \leq L)$, $SYN(MON, = L)$ and $k\text{-}SYN(MON, = L)$ for $k \geq 1$ can be solved in time $O(n^2k)$, i.e. in polynomial time. But there are problems concerning synchronization of the monotonic DFA which cannot be solved in polynomial time (if $P \neq NP$).

In the proof of the next proposition we use a token model of synchronization. Let $\mathscr{A} = (Q, \Sigma, \delta)$ be a DFA and $w \in \Sigma^*$. Suppose that at the beginning there is a token on any state from Q. We apply letters of the word w step by step. The action of the letter $a \in \Sigma$ moves the token from the state $q \in Q$ to the state $\delta(q, a)$. If two tokens arrive at one state, then one of them must be removed. If after the action of the word w there is only one token on the set Q, then the word w is a reset word. If after the action of the word w there are M tokens on the states of the set Q, then the word w compresses the automaton $\mathscr{A}$ to M states.

Proposition 7.1. *1. The problems $COMP(MON, M, \leq L)$ and $k\text{-}COMP(MON, M, \leq L)$ for $k \geq 2$ are NP-complete.*

2. The problems $COMP(MON, M, = L)$ and $k\text{-}COMP(MON, M, = L)$ for $k \geq 2$ are NP-hard and co-NP-hard.

Proof. It can be checked in polynomial time, whenever a given word compresses a given DFA to M states. Hence, the problem $COMP(MON, M, \leq L)$ belongs to NP. If we prove the NP-hardness of the problem $2\text{-}COMP(MON, M, \leq L)$ then the NP-completeness of the problems $COMP(MON, M, \leq L)$ and $k\text{-}COMP(MON, M, \leq L)$ for $k \geq 2$ will be proved as well.

We reduce the problem SAT to the problem $2\text{-}COMP(MON, M, \leq L)$. Let the input of the problem SAT is a set of clauses $c_1(x_1, \ldots, x_n), \ldots, c_p(x_1, \ldots, x_n)$ over the variables $x_1, \ldots, x_n$. We are going to construct a 2-letter automaton $\mathscr{A}_{mon} = (Q, \{a, b\}, \delta)$ and the numbers M and L such that there exists a word of length L compressing the automaton $\mathscr{A}_{mon}$ to the M states if and only if there exist values of variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = \ldots = c_p(x_1, \ldots, x_n) = 1$.

Let $\Sigma = \{a, b\}$, $Q = \{q(m', i) | i \in \{1, \ldots, p\}, m' \in \{1, 2n + 2\}\}$. Let $i \in \{1, \ldots, p\}, m \in \{1, \ldots, n\}$, then

$$q(2m-1, i).a = \begin{cases} q(2m+2, i), & \text{if } x_m \text{ is contained in } c_i \text{ without } \neg \\ q(2m+1, i), & \text{otherwise} \end{cases}$$

$$q(2m-1, i).b = \begin{cases} q(2m+2, i), & \text{if } \neg x_m \text{ is contained in } c_i \\ q(2m+1, i), & \text{otherwise} \end{cases}$$

$$q(2m, i).a = q(2m+2, i), q(2m, i).b = q(2m+2, i).$$

$$q(2n+1, i).a = q(2n+1, i).b = q(2n+2, i).a = q(2n+2, i).b = q(2n+2, i).$$

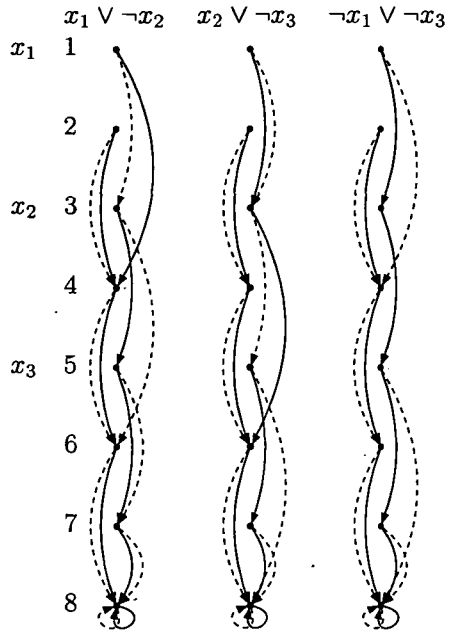


Figure 6: Automaton $\mathscr{A}_{mon}$ for clauses $x_1 \vee \neg x_2$, $x_2 \vee \neg x_3$, $\neg x_1 \vee \neg x_3$

We also put $M = p$, $L = n$. An example of the automaton $\mathscr{A}_{mon}$ is represented by Figure 6. The action of the letter a is denoted with solid lines. The action of the letter b is denoted with dotted lines. The figure contains three columns of states. In the i-th column there are states of kind $q(m, i)$ for fixed i. In any horizontal row there are states $q(m, i)$ for some fixed m.

We define a linear order $\leq$ on the set Q. We put

$$q(m_1, i_1) \leq q(m_2, i_2), \text{ if } i_1 < i_2, \text{ or } i_1 = i_2 \text{ and } m_1 \leq m_2.$$

It is not difficult to verify that for each letter $a \in \Sigma$ the transformation $\delta(\_, a)$ of the set Q preserves $\leq$. Thus, the automaton $\mathscr{A}_{mon}$ is monotonic. The size of the automaton $\mathscr{A}_{mon}$ is a polynomial in common number of clauses and variables.

The set Q can be represented as a table with p columns and $2n + 2$ rows. In the i-th column K_i there are states of kind $q(*, i)$, in the m-th row R_m there are states of kind $q(m, *)$. Suppose that there is a token on every state of the set Q at the start of the synchronization. If some word compresses the automaton $\mathscr{A}_{mon}$ to p states, then it moves all tokens to the states $q(2n + 2, 1), \ldots, q(2n + 2, p)$, i.e. to the $2n + 2$-th row. Let some token be in the row R_m, $m \in \{1, \cdots, 2n\}$. This token can be moved to the row R_{m+2} or to the row R_{m+3} under the action of some letter. Thus, if $q \in R_2 \cup \ldots \cup R_{2n+2}$ and $w \in \Sigma^*$, $|w| = n$, then $q.w \in R_{2n+2}$. Therefore, a word w of length n compresses the automaton $\mathscr{A}_{mon}$ if and only if $R_1.w = R_{2n+2}$.

Let there exist values of the variables $x_1, \ldots, x_n$ such that $c_1(x_1, \ldots, x_n) = \ldots = c_p(x_1, \ldots, x_n) = 1$. Consider a word w of length n such that

$w[m] = \begin{cases} a, & \text{if } x_m = 1 \\ b, & \text{if } x_m = 0 \end{cases}$ for $m \in \{1,\dots,n\}$. Let $i \in \{1,\dots,p\}$. Let m be a minimal number from the set $\{1,\dots,n\}$ such that $x_m = 1$ and x_m is contained in c_i without $\neg$, or $x_m = 0$ and $\neg x_m$ is contained in c_i. Then

$$q(1,i).w[1] = q(3,i), q(3,i).w[2] = q(5,i)\dots q(2m-3,i).w[m-1] = q(2m-1,i)$$

$$q(2m-1,i).w[m] = q(2m+2,i)$$

$$q(2m+2,i).w[m+1] = q(2m+4,i),\dots q(i,2n).w[n] = q(i,2n+2)$$

Therefore, $R_1.w = R_{2n+2}$ and $|Q.w| = p = M$.

Let there exist a word $w \in \Sigma^*$ of length n such that $|Q.w| = p$. In this case $Q.w = \{q(2n+2,1),\dots,q(2n+2,p)\}$. We put $x_m = \begin{cases} 1, & \text{if } w[m] = a \\ 0, & \text{if } w[m] = b \end{cases}$. Let $i \in \{1,\dots,p\}$, then $q(1,i).w = q(2n+2,i)$. Let us consider a token from the state $q(1,i)$. If each letter of the word w moves this token from row with number j to row with number $j+2$, then after applying the word w the token cannot be on the state $q(2n+2,i)$. Therefore, there is an $m \in \{1,\dots,n\}$ such that $q(2m-1,i).w[m] = q(2m+2,i)$. This holds only if the variable x_m is contained in c_i without $\neg$ and $x_m = 1$; or $\neg x_m$ is contained in c_i and $x_m = 0$. In this case $c_i(x_1,\dots,x_n) = 1$.

2. The statement can be proved using the idea of the proof of the NP and co-NP-hardness of the problem 2-$SYN(DFA)$. But in this case idea should be applied to the automaton $\mathscr{A}_{mon}$. □

Acknowledgement

The author is grateful to his supervisor Dr. D.S. Ananichev for suggesting the research problem and for his valuable help. The author acknowledges support from the Federal Education Agency of Russia, grant 2.1.1/3537, and from the Russian Foundation for Basic Research, grant 09-01-12142.

References

[1] Černý, J. Poznámka k homogénnym eksperimentom s konecnými avtomatami. *Mat.-Fyz. Cas. Slovensk. Akad. Vied.*, 14:208–216, 1964 [in Slovak].

[2] Eppstein, D. Reset sequences for monotonic automata. *SIAM J. Comput.*, 19:500–510, 1990.

[3] Salomaa, A. Composition sequences for functions over a finite domain. *Theor. Comput. Sci.*, 292:263–281, 2003.

[4] Samotij, W. A note on the complexity of the problem of finding shortest synchronizing words. Palermo, AUTOMATA 2007.

[5] Martugin, P. A series of slowly synchronizable automata with a zero state over a small alphabet. *Inf. and Comput.*, 206:1197–1203, 2008.

[6] Ananichev, D. S. and Volkov, M. V. Synchronizing monotonic automata. *Theoret. Comput. Sci.*, 327:225–239, 2004.

[7] Ananichev, D. S. and Volkov, M. V. Synchronizing generalized monotonic automata. *Theoret. Comput. Sci.*, 330:3–13, 2005.

[8] Ananichev, D. S. The mortality threshold for partially monotonic automata. In de Felice, C. and Restivo, A., editors, *Developments in Language Theory, 9th International Conference, DLT 2005, Palermo, Italy, July 4-8, 2005, Proceedings.*, Lecture Notes in Computer Science 3572, pages 112–121. Springer, 2005.

[9] Trahtman, A. N. The Černý conjecture for aperiodic automata. *Discr. Math. and Theoret. Comput. Sci.*, 9(2):3–10, 2007.

[10] Rystsov, I. C. Reset words for commutative and solvable automata. *Theoret. Comput. Sci.*, 172:273–279, 1997.

[11] Rystsov, I. C. Reset words for automata with simple idempotents. *Kibernetika i Sistemnyj Analiz*, 3:32–39, 2000, [in Russian; English translation: *Cybernetics and System Analysis*, 36:339–344, 2000]

[12] Kari, J. A counter example to a conjecture concerning synchronizing words in finite automata. *EATCS Bull.*, 73:146, 2001.

[13] Kari, J. Synchronizing finite automata on Eulerian digraphs. *Theoret. Comput. Sci.*, 295:223–232, 2003.

[14] Dubuc, L. Sur les automates circulaires et la conjecture de Černý. *RAIRO Inform. Theor. Appl.*, 32:21–34, 1998 [in French].

Received 30th September 2008

Acta Cybernetica **19** (2009) 537–552.

On Pure Multi-Pushdown Automata that Perform Complete Pushdown Pops

Tomáš Masopust* and Alexander Meduna*

Abstract

This paper introduces and discusses pure multi-pushdown automata that remove symbols from their pushdowns only by performing complete pushdown pops. This means that during a pop operation, the entire pushdown is compared with a prefix of the input, and if they match, the whole contents of the pushdown is erased and the input is advanced by the prefix. The paper proves that these automata define an infinite hierarchy of language families identical with the infinite hierarchy of language families resulting from right linear simple matrix grammars. In addition, this paper discusses some other extensions of these automata with respect to operations they can perform with their pushdowns. More specifically, it discusses pure multi-pushdown automata that perform complete pushdown pops that are allowed to join two pushdowns and/or create a new pushdown.

Keywords: pure multi-pushdown automaton, complete pushdown pop, infinite hierarchy.

1 Introduction

Indisputably, *pushdown automata* fulfill a crucial role in formal language theory. Therefore, it comes as no surprise that this theory has introduced many variants of these automata (consult, for example, [1, 5, 6, 7, 8, 11, 12, 14, 17, 18] for more details).

It is well-known that the family of languages accepted by pushdown automata (with only one pushdown) coincides with the family of context-free languages and that adding any more pushdown makes these automata as powerful as Turing machines. Considering the pushdown alphabet, it is not hard to see that any number of pushdown symbols can be encoded by two different pushdown symbols. However, if the pushdown alphabet is a singleton (more precisely, we have one pushdown symbol A, and a bottom-of-pushdown symbol Z, $Z \neq A$, Z appears only on the bottom of the pushdown), we obtain so-called *counter automata* or *counter machines*. It is known that these automata accept languages from a proper subfamily of the family of context-free languages if they are equipped with only one counter, or the family of recursively enumerable languages if they are equipped

*Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 61266, Czech Republic, E-mail: tomas.masopust@mail.muni.cz, meduna@fit.vutbr.cz.

with two or more counters (see [9]). Furthermore, the pushdown alphabet can, in general, contain symbols that are not in the input alphabet, i.e., symbols that will never occur as a part of the input. If the pushdown automata are restricted so that the pushdown alphabet does not contain any such symbols, we obtain so-called *pure pushdown automata*. Clearly, with respect to the cardinality of the input alphabet, pure pushdown automata with only one input symbol are as powerful as counter automata, whereas with two or more different input symbols they are as powerful as pushdown automata. Therefore, the family of languages accepted by pure pushdown automata with only one pushdown coincides either with the family of languages accepted by one-counter automata, or with the family of context-free languages. Finally, it immediately follows from the previous explanations that pure pushdown automata with two or more pushdowns are as powerful as Turing machines. For an overview of multi-pushdown automata see the paper by Fischer [4] and the references therein.

The present paper continues the investigations in this classical topic of formal language theory. More specifically, it discusses pure multi-pushdown automata that can remove symbols from their pushdowns only by performing a complete pushdown pop. This means that during a pop operation, the entire pushdown is compared with a prefix of the input, and if they match, the whole contents of the pushdown is eliminated and, simultaneously, the input is advanced by the prefix. This paper demonstrates that these automata define an infinite hierarchy of language families identical with the infinite hierarchy of language families resulting from the following grammars and automata:

1. equal matrix languages (see Siromoney [16]);
2. right linear simple matrix grammars (see Ibarra [10]);
3. multi-tape one-way non-writing automata (see Fischer and Rosenberg [5]);
4. finite-turn checking automata (see Siromoney [17]);
5. all-move self-regulating finite automata (see Meduna and Masopust [13]).

In addition, this paper discusses pure multi-pushdown automata that perform complete pushdown pops that are allowed (in some sense) to join two pushdowns and/or introduce a new pushdown. These operations imply another infinite hierarchy of language families dependent upon the number of pushdowns.

In its conclusion, this paper formulates some open problems.

2 Preliminaries and Definitions

In this paper, we assume that the reader is familiar with the theory of automata and formal languages (see [15]). For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$. For $w \in V^*$ and $W \subseteq V$, w^R denotes the mirror image of w and $occur(w, W)$ denotes the number of occurrences of symbols from W in w. Let $\mathscr{L}_{REG}$ denote the family of regular languages.

A *context-free grammar* is a quadruple $G = (N,T,P,S)$, where N is a nonterminal alphabet, T is a terminal alphabet such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$ is the start symbol, and P is a finite set of productions of the form $A \to v$, where $A \in N$ and $v \in V^*$.

In what follows, productions from P are labeled by elements of a finite set Q chosen so that there is a bijection *lab* from P to Q. Then, $Q = lab(P) = \{lab(p) : p \in P\}$ is said to be a set of *production labels*. For the brevity, we hereafter write $q : A \to v \in P$ instead of $A \to v \in P$ with $lab(A \to v) = q$.

Let $q : A \to v \in P$ and $x,y \in V^*$. Then, G makes a derivation step from xAy to xvy, written as $xAy \Rightarrow xvy$. In the standard way, we define $\Rightarrow^m$, for $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. To express that G performs $x \Rightarrow^m y$, for some $x,y \in V^*$, by using a sequence of productions $q_1,q_2,\ldots,q_m$, we write $x \Rightarrow^m y\ [q_1q_2\ldots q_m]$. The language generated by a context-free grammar G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$ and is said to be a *context-free language*. The family of all context-free languages is denoted by $\mathscr{L}_{CF}$.

For $n \geq 1$, an *n-right linear simple matrix grammar* (defined in [16] (as equal matrix grammars) and in [10], see also [19]) is an $(n+3)$-tuple $G = (N_1,N_2,\ldots,N_n,T,P,S)$, where $N_1,N_2,\ldots,N_n$ are pairwise disjoint nonterminal alphabets, T is a terminal alphabet, $N = N_1 \cup N_2 \cup \cdots \cup N_n$, $S \notin N \cup T$ is the start symbol, $N \cap T = \emptyset$, and P is a finite set of matrix productions of the following three forms:

1. $[S \to X_1X_2\ldots X_n]$, $X_i \in N_i$, $1 \leq i \leq n$;
2. $[X_1 \to w_1Y_1, X_2 \to w_2Y_2, \ldots, X_n \to w_nY_n]$, $w_i \in T^*$, $X_i,Y_i \in N_i$, $1 \leq i \leq n$;
3. $[X_1 \to w_1, X_2 \to w_2, \ldots, X_n \to w_n]$, $X_i \in N_i$, $w_i \in T^*$, $1 \leq i \leq n$.

For $x,y \in (N \cup T \cup \{S\})^*$, $x \Rightarrow y$ provided that

1. either $x = S$ and $[S \to y] \in P$,
2. or $x = y_1X_1y_2X_2\ldots y_nX_n$, $y = y_1x_1y_2x_2\ldots y_nx_n$, and $[X_1 \to x_1,\ldots,X_n \to x_n] \in P$.

As usual, we define $\Rightarrow^m$, for $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by an n-right linear simple matrix grammar G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$ and is said to be an *n-right linear simple matrix language*. The family of all n-right linear simple matrix languages is denoted by $\mathscr{L}_R^n$.

A *programmed grammar* is a quadruple $G = (N,T,P,S)$, where N is a nonterminal alphabet, T is a terminal alphabet such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$ is the start symbol, and P is a finite set of productions of the form $(q : A \to v, g(q))$, where $q : A \to v$ is a labeled context-free production and $g(q) \subseteq lab(P)$.

In every derivation of G, any two consecutive steps, $x \Rightarrow y \Rightarrow z$, made by productions $(p : A \to u, g(p))$ and $(q : B \to v, g(q))$, respectively, satisfy $q \in g(p)$. As usual, we define $\Rightarrow^m$, for $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language generated by a programmed grammar G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$ and is said to be a *programmed language*. The family of all programmed languages is denoted by $\mathscr{L}_P$.

Let D be a derivation of a string $w \in V^*$ in G of the form $w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_r$, for some $r \geq 1$, where $S = w_1$ and $w_r = w$. Set $Ind(D,G) = \max\{occur(w_i,N) : 1 \leq i \leq r\}$. For $w \in T^*$, set $Ind(w,G) = \min\{Ind(D,G) : D \text{ is a derivation of } w \text{ in } G\}$. The *index* of G is defined as $Ind(G) = \max\{Ind(w,G) : w \in L(G)\}$.

For $L \in \mathscr{L}_P$, set $Ind(L) = \min\{Ind(G) : L(G) = L, G \text{ is a programmed grammar}\}$. Finally, let $\mathscr{L}_P^n = \{L \in \mathscr{L}_P : Ind(L) \leq n\}$, for all $n \geq 1$, denote the family of all *programmed languages of index n*.

2.1 Pure Multi-Pushdown Automata that Perform Complete Pushdown Pops

Let n be a positive integer. A *pure n-pushdown automaton that performs complete pushdown pops*, an nPPDA for short, is a quadruple

$$M = (Q, T, R, s),$$

where Q is a finite set of states, T is an alphabet of input symbols, $R \subseteq \mathscr{S} \times \mathscr{S}$ is a set of rules, $\mathscr{S} = \mathscr{S}_1 \cup \mathscr{S}_2 \cup \mathscr{S}_3 \cup \mathscr{S}_4$,

- $\mathscr{S}_1 = \{\langle q, pop\rangle : q \in Q\}$
- $\mathscr{S}_2 = \{\langle q, push, i, a\rangle : q \in Q, 1 \leq i \leq n, a \in T \cup \{\varepsilon\}\}$
- $\mathscr{S}_3 = \{\langle q, new, i\rangle : q \in Q, 1 \leq i \leq n\}$
- $\mathscr{S}_4 = \{\langle q, join, i\rangle : q \in Q, 2 \leq i \leq n\}$

and $s \notin \mathscr{S}$ is the start state. In what follows, we use the notation $p \to q$ for $(p, q) \in R$.

A configuration of M is a string over

$$(T^*\{\$\} \cup \{\varepsilon\})^n \times (\mathscr{S} \cup \{s\}) \times T^*.$$

Let $1 \leq k \leq n$ and $p \to q \in R$. We define the relation $\Rightarrow$ depending on the left-hand side of $p \to q$, i.e., p, as follows:

1. $\$^n sw \Rightarrow \$^n qw$, for $p = s$;
2. $w_k\$ \ldots \$w_2\$w_1\$pw_1^R w \Rightarrow w_k\$ \ldots \$w_2\$qw$ for $p = \langle r, pop\rangle$;
3. $w_k\$ \ldots \$w_i\$ \ldots \$w_1\$pw \Rightarrow w_k\$ \ldots \$w_i a\$ \ldots \$w_1\$qw$, for $p = \langle r, push, i, a\rangle$ and $i \leq k$;
4. $w_k\$ \ldots \$w_i\$ \ldots \$w_1\$pw \Rightarrow w_k\$ \ldots \$w_i\$\$ \ldots \$w_1\$qw$, for $p = \langle r, new, i\rangle$ and $i \leq k < n$;
5. $w_k\$ \ldots \$w_1\$pw \Rightarrow \$w_k\$ \ldots \$w_1\$qw$, for $p = \langle r, new, k+1\rangle$ and $k < n$;
6. $w_k\$ \ldots \$w_i\$w_{i-1}\$ \ldots \$w_1\$pw \Rightarrow w_k\$ \ldots \$w_i w_{i-1}\$ \ldots \$w_1\$qw$, for $p = \langle r, join, i\rangle$ and $i \leq k$.

Remark 1. Note that symbols \$ denote the tops of M's pushdowns and that the automaton cannot make a computational step unless there is at least one pushdown.

In the standard way, we define $\Rightarrow^m$, for $m \geq 0$, and $\Rightarrow^*$. Then, the language of an nPPDA M is defined as

$$L(M) = \{w \in T^* : \$^n sw \Rightarrow^* q, \text{ for some } q \in \mathscr{S}\},$$

where $\$^n sw \Rightarrow^* q$ is said to be a *successful computation of M on w*.

Finally, for $I \subseteq \{1,2,3,4\}$,

$$\mathscr{L}_I^n = \left\{L(M) : M = (Q,T,R,s) \text{ is an } n\text{PPDA with } R \subseteq \bigcup_{i\in I} \mathscr{S}_i \times \bigcup_{i\in I} \mathscr{S}_i\right\}.$$

3 Main Results

In this section, we demonstrate two infinite language hierarchies generated by pure multi-pushdown automata that perform complete pushdown pops according to their pushdown operations and the number of pushdowns. First, however, we generalize these automata so that they are allowed to push a string to their pushdowns in one computational step instead of only one symbol or the empty string.

3.1 Generalized nPPDAs

A *generalized nPPDA* is an nPPDA $M = (Q,T,R,s)$ with $R \subseteq \mathscr{S}' \times \mathscr{S}'$, where $\mathscr{S}'$ is a finite subset of $\mathscr{S}_1 \cup \mathscr{S}_2' \cup \mathscr{S}_3 \cup \mathscr{S}_4$; sets $\mathscr{S}_1$, $\mathscr{S}_3$, $\mathscr{S}_4$ are as in the case of standard nPPDA, and $\mathscr{S}_2$ is modified so that $a \in T$ is replaced with $u \in T^*$:

- $\mathscr{S}_2' = \{\langle q,push,i,u\rangle : q \in Q, 1 \leq i \leq n, u \in T^*\}$.

Correspondingly, the computational step is modified as follows:

3. $w_k\$\ldots\$w_i\$\ldots\$w_1\$\langle p,push,i,u\rangle w \Rightarrow w_k\$\ldots\$w_i u\$\ldots\$w_1\$qw$, for $i \leq k$.

The other computational steps are defined as in the case of standard nPPDA.

First, we prove that this generalization has no effect to the acceptance power of these automata.

Lemma 1. *Let M be a generalized nPPDA, for some $n \geq 1$. Then, there is an nPPDA, M', such that $L(M) = L(M')$.*

Informally, what M does in one derivation step, M' does in the-length-of-the-added-string steps.

Proof. Let $M = (Q,T,R,s)$ be a generalized nPPDA. Construct the following nPPDA $M' = (Q',T,R',s)$ by the following algorithm ($\mathscr{S}$ is as in the definition in Section 2.1):

1. Set $R' = \{p \to q \in R : p,q \in \mathscr{S} \cup \{s\}\}$ and $Q' = Q$;
2. For each $p \to \langle q,push,i,a_1a_2\ldots a_k\rangle \in R$ with $a_i \in T$, for $i = 1,\ldots,k$, $k \geq 2$, add
   a) states $q_{a_1a_2\ldots a_k}^{i,1}, q_{a_1a_2\ldots a_k}^{i,2}, \ldots, q_{a_1a_2\ldots a_k}^{i,k}$ to Q';

b) $p \to \langle q^{i,1}_{a_1a_2\ldots a_k}, push, i, a_1\rangle$ to R';

c) $\langle q^{i,j}_{a_1a_2\ldots a_k}, push, i, a_j\rangle \to \langle q^{i,j+1}_{a_1\ldots a_k}, push, i, a_{j+1}\rangle$ to R', for $j = 1,\ldots,k-1$;

d) for each $\langle q, push, i, a_1a_2\ldots a_k\rangle \to r \in R$, add
$\langle q^{i,k}_{a_1a_2\ldots a_k}, push, i, a_k\rangle \to r$ to $\begin{cases} R' & \text{for } r \in \mathscr{S}, \\ R & \text{otherwise.} \end{cases}$

3. If R' has been changed, then go to step 2.

It is not hard to see that $L(M) = L(M')$. □

3.2 Language Families

Consider an arbitrary $I \subseteq \{1,2,3,4\}$. It is not hard to see that if $1 \notin I$, then $\mathscr{L}^n_I = \emptyset$; such an automaton cannot remove \$s from its configurations. Furthermore, if $1 \in I$ and $2 \notin I$, then $\mathscr{L}^n_I = \{\varepsilon\}$; of course, such an automaton can remove all symbols \$ but cannot read any nonempty input. Thus, there are only four sets of interest: $\{1,2\}$, $\{1,2,3\}$, $\{1,2,4\}$, $\{1,2,3,4\}$. The following two lemmas are obvious.

Lemma 2. *For all $n \geq 1$,*

1. $\mathscr{L}^n_{\{1,2\}} \subseteq \mathscr{L}^n_{\{1,2,3\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$,
2. $\mathscr{L}^n_{\{1,2\}} \subseteq \mathscr{L}^n_{\{1,2,4\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$.

Lemma 3. $\mathscr{L}^1_{\{1,2\}} = \mathscr{L}^1_{\{1,2,3,4\}} = \mathscr{L}_{REG}$.

Now, consider an automaton with pop, push, and join operations. We will show how the join operation can be simulated by only push and pop operations without any change of the accepted language. Notice that the join operation applied to the ith pushdown appends the content of the ith pushdown to the bottom of the $(i-1)$st pushdown. Thus, to push a symbol to the jth pushdown in this automaton, for some $j \geq i$, equals to skipping the join operation and pushing the symbol to the $(j+1)$st pushdown. This is generalized and done by a sequence of the form $i_1i_2\ldots i_m$ added to states, for some $m \leq n$, where $i_k \in \{1,0\}$, for $k = 1,\ldots,m$, and $i_k = 0$ if and only if the i_kth pushdown has been joined. Then, the automaton starts with a sequence of n 1s, $11\ldots1$, in its start state, and to push a symbol to the ith pushdown means to push the symbol to the lth pushdown, where l is the position of the ith 1 in the sequence from the left. Analogously, to make the pop operation, say from a state with $10\ldots0i_l\ldots i_k$, where $2 \leq l \leq k$ and $i_l = 1$, the new automaton makes $l-1$ pop operations and goes to a state with $i_l\ldots i_k$. Finally, to join the ith pushdown means to replace the ith 1 with 0 in the state by the push operation pushing ε to the first pushdown.

Hence, we have the following lemma.

Lemma 4. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} = \mathscr{L}^n_{\{1,2,4\}}$.*

Corollary 1. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} = \mathscr{L}^n_{\{1,2,4\}} \subseteq \mathscr{L}^n_{\{1,2,3\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$.*

As far as the $\mathscr{L}^n_{\{1,2,3\}}$ language families are concerned, $n \geq 2$, we only know the following result.

Theorem 1. *For all* $n \geq 2$, $\mathscr{L}^n_{\{1,2\}} \subset \mathscr{L}^n_{\{1,2,3\}}$.

Proof. Let $L = \{a^k b^k : k \geq 1\}$. Clearly, $L \in \mathscr{L}^2_{\{1,2\}}$. Ibarra [10, Theorem 4.7] showed that $L^* \notin \mathscr{L}^m_{\{1,2\}}$, for $m \geq 1$. To prove this theorem, we show that $L^* \in \mathscr{L}^2_{\{1,2,3\}}$.

The automaton starts with the initial configuration $\$\$sa^k b^k w$, for some $w \in L^*$. Then, it simultaneously generates a^k and b^k in the first and the second pushdown, respectively, i.e., the configuration is of the form $b^k\$a^k\$pa^k b^k w$, for some $p \in \mathscr{S}$. Then, the automaton pops the first pushdown (reading a^k from the input) and creates a new one on the second position, i.e., its configuration is $\$b^k\$qb^k w$, for some $q \in \mathscr{S}$. Then, it pops the first pushdown again (reading b^k from the input) and creates a new pushdown, i.e., the configuration is $\$\$rw$, for some $r \in \mathscr{S}$. Thus, the cycle can be repeated. The formal proof is left to the reader. □

Corollary 2. $\mathscr{L}^2_{\{1,2,3\}} - \bigcup_{n=1}^{\infty} \mathscr{L}^n_{\{1,2\}} \neq \emptyset$.

In the following two sections, language families $\mathscr{L}^n_{\{1,2\}}$ and $\mathscr{L}^n_{\{1,2,3,4\}}$ are discussed. Note that most of the questions concerning language families $\mathscr{L}^n_{\{1,2,3\}}$ are open (see the last section for more details).

3.3 Language Families $\mathscr{L}^n_{\{1,2\}}$

First, let us give an example. Note that in case $n = 2$, this example shows that the language, L, from the proof of Theorem 1 is in $\mathscr{L}^2_{\{1,2\}}$ as stated there.

Example 1. Consider an nPPDA $M = (\{s,q\}, \{a_1, a_2, \ldots, a_n\}, R, s)$ with R having the following rules:

1. $s \rightarrow \langle q, push, 1, a_1 \rangle$,
2. $\langle q, push, i, a_i \rangle \rightarrow \langle q, push, i+1, a_{i+1} \rangle$, for $i = 1, \ldots, n-1$,
3. $\langle q, push, n, a_n \rangle \rightarrow \langle q, push, 1, a_1 \rangle$,
4. $\langle q, push, n, a_n \rangle \rightarrow \langle q, pop \rangle$,
5. $\langle q, pop \rangle \rightarrow \langle q, pop \rangle$.

Then, $L(M) = \{a_1^k a_2^k \ldots a_n^k : k \geq 1\}$. □

In the following, we prove that the power of nPPDAs with push and pop operations is precisely the power of n-right linear simple matrix grammars. First, however, notice that any such automaton, M, has the property that there is precisely n pop operations in any of its successful computations; clearly, the automaton has to pop n pushdowns and no new pushdown can be created. Moreover, we can prove that there is an equivalent automaton, M', such that in any successful computation of M', no pop operation precedes a push operation. To show this, let M' simulate M but if M pops the pushdown, M' skips the pop operation and increases the number of pop operations skipped so far recorded in its state. Thus, in any time, M' knows the number of pop operations applied in the corresponding

computation of M, say k, $0 \leq k \leq n$. Then, if M pushes a symbol to the ith pushdown, M' pushes this symbol to the $(i+k)$th pushdown. Clearly, M' finishes (pops all its pushdowns one by one) only if M has performed n pop operations.

Lemma 5. *Let $n \geq 1$ and $L \in \mathscr{L}^n_{\{1,2\}}$. Then, there is an nPPDA, M, such that $L(M) = L$ and its sequence of operations applied during any successful computation, starting from s, is of the form*

$$s, push_1, push_2, \ldots, push_k, pop_1, pop_2, \ldots, pop_n$$

for some $k \geq 1$, $push_i \in \mathscr{S}_2$, for all $i = 1, \ldots, k$, and $pop_j \in \mathscr{S}_1$, for all $j = 1, \ldots, n$.

Proof. This immediately follows from the previous arguments and the fact that if there is no push operation in the successful computation, then we can push ε to the first pushdown, i.e., for some state t, $push_1 = \langle t, push, 1, \varepsilon \rangle$. □

Lemma 6. *For all $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} \subseteq \mathscr{L}^n_R$.*

Proof. Let $M = (Q, T, R, s)$ be an nPPDA with $R \subseteq (\mathscr{S}_1 \cup \mathscr{S}_2) \times (\mathscr{S}_1 \cup \mathscr{S}_2)$ satisfying the condition from Lemma 5. Clearly, without loss of generality, we can assume that $pop_1 = pop_2 = \cdots = pop_n = \langle r, pop \rangle$, for some $r \in Q$. Thus, $\mathscr{S}_1 = \{\langle r, pop \rangle\}$.

Let $G = (N_1, \ldots, N_n, T, P, S_G)$ and set $N_i = (\mathscr{S}_1 \cup \mathscr{S}_2) \times \{i\}$, for all $i = 1, \ldots, n$. Set $P = \{S_G \to \langle \langle r, pop \rangle, 1 \rangle \langle \langle r, pop \rangle, 2 \rangle \ldots \langle \langle r, pop \rangle, n \rangle : \langle r, pop \rangle \in \mathscr{S}_1\}$.

If $q \to p \in R$ is of the form

1. $\langle t, push, i, a \rangle \to \langle r, pop \rangle$, add
   $[\langle \langle r, pop \rangle, 1 \rangle \to \langle q, 1 \rangle, \ldots, \langle \langle r, pop \rangle, i \rangle \to a \langle q, i \rangle, \ldots, \langle \langle r, pop \rangle, n \rangle \to \langle q, n \rangle]$ to P;

2. $\langle r, push, i, a \rangle \to \langle t, push, j, b \rangle$, add
   $[\langle p, 1 \rangle \to \langle q, 1 \rangle, \ldots, \langle p, i \rangle \to a \langle q, i \rangle, \ldots, \langle p, n \rangle \to \langle q, n \rangle]$ to P;

3. $s \to p$, add
   $[\langle p, 1 \rangle \to \varepsilon, \ldots, \langle p, i \rangle \to \varepsilon, \ldots, \langle p, n \rangle \to \varepsilon]$ to P.

Note that M starts with $s \to p$, continues with $p \to q$ followed by the application of a rule of the form $p \to \langle r, pop \rangle$, for some $p, q \in \mathscr{S}_2$, and finishes with $\langle r, pop \rangle \to \langle r, pop \rangle$ applied n-times. Denote the sequence of applied rules by $s, p_1, \ldots, p_k, pop_1, \ldots, pop_n$, for some $k \geq 1$. Then, G simulates M by the following sequence of productions: the initial production (simulating all n pop operations) followed by a sequence of productions $p'_k, \ldots, p'_1, s'$, where p'_k is constructed from p_k as in 1, p'_i from p_i as in 2, for all $i = 1, \ldots, k-1$, and s' from s as in 3. □

Lemma 7. *For all $n \geq 1$, $\mathscr{L}^n_R \subseteq \mathscr{L}^n_{\{1,2\}}$.*

Proof. Let $n \geq 1$ and $G = (N_1, \ldots, N_n, T, P, S)$ be an n-right linear simple matrix grammar. Construct the following generalized nPPDA $M = (Q, T, R, s)$, where $Q = \{(x, m) : x \in N_1 \ldots N_n, m \in P\} \cup \{S\}$ and R is defined as follows:

1. For $\alpha = X_1 \ldots X_n \in N_1 \ldots N_n$ and $m = [X_1 \to w_1, \ldots, X_n \to w_n] \in P$ with $w_i \in T^*$, for all $i = 1, 2, \ldots, n$, add $s \to \langle (\alpha, m), push, 1, w_1^R \rangle$ to R;

2. For $\alpha = X_1 \dots X_n$, $\beta = Y_1 \dots Y_n \in N_1 \dots N_n$, and $m' = [Y_1 \to v_1 X_1, \dots, Y_n \to v_n X_n] \in P$, add $\langle (\alpha, m), push, n, w_n^R \rangle \to \langle (\beta, m'), push, 1, v_1^R \rangle$ to R;

3. For $\alpha = Y_1 \dots Y_n \in N_1 \dots N_n$ and $m[i+1] = Y_{i+1} \to v_{i+1} X_{i+1}$ ($m[i]$ denotes the ith element of m) with $v_{i+1} \in T^*$ and $X_{i+1} \in N \cup \{\varepsilon\}$, for all $i = 1, \dots, n-1$, add $\langle (\alpha, m), push, i, v_i^R \rangle \to \langle (\alpha, m), push, i+1, v_{i+1}^R \rangle$ to R;

4. For $\alpha = X_1 \dots X_n$, if there is $[S \to X_1 \dots X_n] \in P$, add $\langle (\alpha, m), push, n, v_n^R \rangle \to \langle S, pop \rangle$ to R;

5. Add $\langle S, pop \rangle \to \langle S, pop \rangle$ to R.

Clearly, M simulates the derivation of G bottom-up and what G does in one derivation step, M does in n steps. Then, according to Lemma 1, the proof is complete. □

The following theorem presents the main result of this section.

Theorem 2. *For all* $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} = \mathscr{L}^n_R$.

Proof. This immediately follows from the previous two lemmas. □

Corollary 3. *For all* $n \geq 1$, $\mathscr{L}^n_{\{1,2\}} \subset \mathscr{L}^{n+1}_{\{1,2\}}$.

Proof. This follows from the previous theorem and Theorem 2.3 in [10]. □

3.4 Language Families $\mathscr{L}^n_{\{1,2,3,4\}}$

The following lemma shows that any language accepted by an nPPDA can be generated by a programmed grammar of index $n+1$.

Lemma 8. *For all* $n \geq 1$, $\mathscr{L}^n_{\{1,2,3,4\}} \subseteq \mathscr{L}^{n+1}_P$.

Before the formal proof of the lemma, we provide some explanations to the construction. Informally, to an nPPDA M, we construct a programmed grammar, G, of index $n+1$ so that the ith nonterminal of G, which is of the form $\langle A_i, k \rangle$, $1 \leq k \leq n+1$, is associated with the ith pushdown. Specifically, if the current content of M's pushdowns is $c_2 c_1 \$ b_2 b_1 \$ a_2 a_1 \$$ (corresponding to a string $a_1 a_2 b_1 b_2 c_1 c_2$), then the sentential form of G is of the form $\langle A_1, 4 \rangle a_1 a_2 \langle A_2, 4 \rangle b_1 b_2 \langle A_3, 4 \rangle c_1 c_2 \langle A_4, 4 \rangle$. Then, the pop operation is simulated so that

$$\langle A_1, 4 \rangle a_1 a_2 \langle A_2, 4 \rangle b_1 b_2 \langle A_3, 4 \rangle c_1 c_2 \langle A_4, 4 \rangle$$

is replaced with

$$a_1 a_2 \langle A_1, 3 \rangle b_1 b_2 \langle A_2, 3 \rangle c_1 c_2 \langle A_3, 3 \rangle .$$

The push operation pushing a onto the second pushdown, i.e., $c_2 c_1 \$ b_2 b_1 a \$ a_2 a_1 \$$ corresponding to a string $a_1 a_2 a b_1 b_2 c_1 c_2$, is simulated by replacing

$$\langle A_1, 4 \rangle a_1 a_2 \langle A_2, 4 \rangle b_1 b_2 \langle A_3, 4 \rangle c_1 c_2 \langle A_4, 4 \rangle$$

with

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle ab_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle .$$

The operation introducing a new, say the first, pushdown, i.e., $c_2c_1\$b_2b_1\$a_2a_1\$\$$, is simulated by replacing

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle b_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle$$

with

$$\langle A_1,5\rangle\langle A_2,5\rangle a_1a_2\langle A_3,5\rangle b_1b_2\langle A_4,5\rangle c_1c_2\langle A_5,5\rangle .$$

Note that the previous first pushdown is the second from now on (till the other change). Finally, the join operation of the first and the second pushdown (by a state of the form $\langle r,join,2\rangle$), i.e., $c_2c_1\$b_2b_1a_2a_1\$$, is simulated by replacing

$$\langle A_1,4\rangle a_1a_2\langle A_2,4\rangle b_1b_2\langle A_3,4\rangle c_1c_2\langle A_4,4\rangle$$

with

$$\langle A_1,3\rangle a_1a_2b_1b_2\langle A_2,3\rangle c_1c_2\langle A_3,3\rangle .$$

The formal proof of Lemma 8 follows.

Proof. Let $M=(Q,T,R,s)$ be an nPPDA. Construct the following programmed grammar $G=(N,T,P,S)$, where $N=Q\times\{1,\ldots,n+1\}$ and P is constructed as follows.

Set $f(r)=\{t : r\to t\in R\}$, and $g(f(r))=\bigcup_{p\in f(r)} g(p)$ (the definition of $g(p)$ follows).

1. For any rule $s\to p\in R$, add
   a) $(S\to\langle A_1,n+1\rangle\langle A_2,n+1\rangle\ldots\langle A_{n+1},n+1\rangle, g(p))$ into P;
2. For all $p\in\mathscr{S}_1$ and $1\le l\le n+1$, add
   a) $([\mathbf{p},l,p] : \langle A_1,l\rangle\to\varepsilon,\{[/,2,l,p] : [/,2,l,p]\in lab(P)\})$;
3. For all $p\in\mathscr{S}_1\cup\mathscr{S}_4$ and $1\le i,l\le n+1$, $i\ge 2$, add
   a) $([/,i,l,p] : \langle A_i,l\rangle\to\langle A_{i-1},l\rangle,\{[/,i+1,l,p]\})$, for $i<l$;
   b) $([/,l,l,p] : \langle A_l,l\rangle\to\langle A_{l-1},l\rangle,\{[-,1,l,p]\})$;
4. For all $p\in\mathscr{S}_1\cup\mathscr{S}_4$ and $1\le i,l\le n+1$, $l\ge 2$, add
   a) $([-,i,l,p] : \langle A_i,l\rangle\to\langle A_i,l-1\rangle,\{[-,i+1,l,p]\})$, for $i<l-1$;
   b) $([-,l-1,l,p] : \langle A_{l-1},l\rangle\to\langle A_{l-1},l-1\rangle, g(f(p)))$;
5. For all $p\in\mathscr{S}_2$ and $1\le i,l\le n+1$, add
   a) $([i,l,p] : \langle A_i,l\rangle\to\langle A_i,l\rangle a, g(f(p)))$;
6. For all $p\in\mathscr{S}_3$ and $1\le i,l\le n+1$, $i\le n$, add
   a) $([*,i,l,i,p] : \langle A_i,l\rangle\to\langle A_{i+1},l\rangle,\{[*,i+1,l,i,p]\})$, for $i<l$;
   b) $([*,l,l,i,p] : \langle A_l,l\rangle\to\langle A_{l+1},l\rangle,\{[\mathbf{n},i+1,l,p]\})$;

7. For all $p \in \mathscr{S}_3$, $1 \leq l \leq n+1$ and $1 < i \leq n+1$, add

   a) $([\mathbf{n},i,l,p] : \langle A_i,l\rangle \to \langle A_{i-1},l\rangle\langle A_i,l\rangle, \{[+,1,l,p]\})$;

8. For all $p \in \mathscr{S}_3$ and $1 \leq i,l < n+1$, add

   a) $([+,i,l,p] : \langle A_i,l\rangle \to \langle A_i,l+1\rangle, \{[+,i+1,l,p]\})$, for $i < l+1$;

   b) $([+,l+1,l,p] : \langle A_{l+1},l\rangle \to \langle A_{l+1},l+1\rangle, g(f(p)))$;

9. For all $p \in \mathscr{S}_4$ and $1 \leq i,l \leq n+1$, add

   a) $([\mathbf{j},i,l,p] : \langle A_i,l\rangle \to \varepsilon, W)$, $W = \{[/,i+1,l,p]\})$ if $i < l$, $W = \{[-,1,l,p]\}$ otherwise.

$g(p)$ depends on p as follows:

$p = \langle r,\textbf{\textit{pop}}\rangle$: $g(p) = \{[\mathbf{p},l,p] : [\mathbf{p},l,p] \in lab(P)\}$;

$p = \langle r,\textbf{\textit{push}},i,a\rangle$: $g(p) = \{[i,l,p] : [i,l,p] \in lab(P)\}$;

$p = \langle r,\textbf{\textit{new}},i\rangle$: $g(p) = \{[*,i,l,i,p] : [*,i,l,i,p] \in lab(P)\}$;

$p = \langle r,\textbf{\textit{join}},i\rangle$: $g(p) = \{[\mathbf{j},i,l,p] : [\mathbf{j},i,l,p] \in lab(P)\}$.

Consider a configuration $w_k\$\ldots\$w_2\$w_1\$pw$ of M and the corresponding sentential form of G, i.e., $(\langle A_1,k+1\rangle w_1\langle A_2,k+1\rangle w_2\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, g(p))$. If $p = \langle r,pop\rangle$, G simulates the computational step as follows:

$$\begin{aligned}
& (\langle A_1,k+1\rangle w_1\langle A_2,k+1\rangle w_2\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, \{[\mathbf{p},k+1,p]\}) \\
\Rightarrow\ & (w_1\langle A_2,k+1\rangle w_2\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, \{[/,2,k+1,p]\}) \\
\Rightarrow^k\ & (w_1\langle A_1,k+1\rangle w_2\ldots\langle A_{k-1},k+1\rangle w_k\langle A_k,k+1\rangle, \{[-,1,k+1,p]\}) \\
\Rightarrow^k\ & (w_1\langle A_1,k\rangle w_2\ldots\langle A_{k-1},k\rangle w_k\langle A_k,k\rangle, g(f(p))).
\end{aligned}$$

If $p = \langle r,push,i,a\rangle$, G simulates the computational step as follows:

$$\begin{aligned}
& (\langle A_1,k+1\rangle w_1\ldots\langle A_i,k+1\rangle w_i\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, \{[i,k+1,p]\}) \\
\Rightarrow\ & (\langle A_1,k+1\rangle w_1\ldots\langle A_i,k+1\rangle aw_i\ldots\langle A_k,k+1\rangle w_k\langle A_{k+1},k+1\rangle, g(f(p))).
\end{aligned}$$

If $p = \langle r,new,i\rangle$, G simulates the computational step as follows:

$$\begin{aligned}
& (\langle A_1,k+1\rangle w_1\ldots w_{i-1}\langle A_i,k+1\rangle w_i\ldots w_k\langle A_{k+1},k+1\rangle, \{[*,i,k+1,i,p]\}) \\
\Rightarrow^{k-i+1}\ & (\ldots w_{i-1}\langle A_{i+1},k+1\rangle w_i\ldots w_k\langle A_{k+2},k+1\rangle, \{[\mathbf{n},i+1,k+1,p]\}) \\
\Rightarrow\ & (\ldots w_{i-1}\langle A_i,k+1\rangle\langle A_{i+1},k+1\rangle w_i\ldots w_k\langle A_{k+2},k+1\rangle, \{[+,1,k+1,p]\}) \\
\Rightarrow^{k+2}\ & (\langle A_1,k+2\rangle w_1\ldots w_k\langle A_{k+2},k+2\rangle, g(f(p))).
\end{aligned}$$

If $p = \langle r,join,i\rangle$, G simulates the computational step as follows:

$$\begin{aligned}
& (\langle A_1,k+1\rangle w_1\ldots w_{i-1}\langle A_i,k+1\rangle w_i\ldots w_k\langle A_{k+1},k+1\rangle, \{[\mathbf{j},i,k+1,p]\}) \\
\Rightarrow\ & (\ldots\langle A_{i-1},k+1\rangle w_{i-1}w_i\langle A_{i+1},k+1\rangle w_{i+1}\ldots, \{[/,i+1,k+1,p]\}) \\
\Rightarrow^{k-i}\ & (\ldots\langle A_{i-1},k+1\rangle w_{i-1}w_i\langle A_i,k+1\rangle w_{i+1}\ldots w_k\langle A_k,k+1\rangle, \{[-,1,k+1,p]\}) \\
\Rightarrow^k\ & (\langle A_1,k\rangle w_1\ldots\langle A_{i-1},k\rangle w_{i-1}w_i\langle A_i,k\rangle\ldots w_k\langle A_k,k\rangle, g(f(p))).
\end{aligned}$$

As any derivation of G simulates a computation of M, we have $L(M) = L(G)$. □

The next lemma shows that any language generated by a programmed grammar of index n is accepted by an $(n+1)$PPDA.

Lemma 9. *For all* $n \geq 1$, $\mathscr{L}_P^n \subseteq \mathscr{L}_{\{1,2,3,4\}}^{n+1}$.

The main idea of the proof is to simulate a derivation of a programmed grammar, G, of index n by a generalized $(n+1)$PPDA, M, so that what G generates to the right of the rewritten nonterminal, say $Aw_1Bw_2Cw_3 \Rightarrow Aw_1B'uw_2Cw_3$, M pushes to its corresponding pushdown, $w_3^R\$w_2^Ru^R\$w_1^R\$$. If G generates a string, v, to the left of the rewritten nonterminal, say $Aw_1Bw_2Cw_3 \Rightarrow Aw_1vB''w_2Cw_3$, then M creates a new pushdown just before the pushdown corresponding to the rewritten nonterminal, $w_3^R\$w_2^R\$\$w_1^R\$$, pushes v^R to the new pushdown, $w_3^R\$w_2^R\$v^R\$w_1^R\$$, and joins the two pushdowns, $w_3^R\$w_2^R\$v^Rw_1^R\$$. By this, M puts v^R to the bottom of the pushdown. In case of the first pushdown, the join operation is replaced with the pop operation. The formal proof follows.

Proof. Let $G = (N,T,P,S)$ be a programmed grammar of index n, for some $n \geq 1$. Construct a generalized $(n+1)$PPDA $M = (Q,T,R,s)$ as follows.

1. Set $Q = (lab(P) \cup \{+\}) \times \bigcup_{k \leq n} N^k \times \{0,1,\ldots,m+1\}$, for $m = \max\{k : A \to u \in P, occur(u,N) = k\}$;

2. For all $p : A \to u_1B_1u_2B_2 \ldots u_kB_ku_{k+1} \in P$, where $u_i \in T^*$ and $B_j \in N$, for all $i = 1,\ldots,k+1$, $j = 1,\ldots,k$, $k \geq 0$, and for all $\langle +, \alpha A\beta, 0\rangle \in Q$, where $\alpha, \beta \in N^*$, and $l = occur(\alpha A, N)$, add the following to R:

   - $s \to \langle\langle +, S, 0\rangle, push, 1, \varepsilon\rangle$,
   - $\langle\langle +, \alpha A\beta, 0\rangle, push, 1, \varepsilon\rangle \to \langle\langle p, \alpha B_1 \ldots B_k\beta, k+1\rangle, push, l+k-1, u_{k+1}^R\rangle$,
   - $\langle\langle p, \alpha B_1 \ldots B_k\beta, k+1\rangle, push, l+k-1, u_{k+1}^R\rangle \to$
     $\langle\langle p, \alpha B_1 \ldots B_k\beta, k\rangle, push, l+k-2, u_k^R\rangle$,
   - $\langle\langle p, \alpha B_1 \ldots B_k\beta, k\rangle, push, l+k-2, u_k^R\rangle \to$
     $\langle\langle p, \alpha B_1 \ldots B_k\beta, k-1\rangle, push, l+k-3, u_{k-1}^R\rangle$,

     $\vdots$

   - $\langle\langle p, \alpha B_1 \ldots B_k\beta, 2\rangle, push, l, u_2^R\rangle \to \langle\langle p, \alpha B_1 \ldots B_k\beta, 1\rangle, new, l\rangle$,
   - $\langle\langle p, \alpha B_1 \ldots B_k\beta, 1\rangle, new, l\rangle \to \langle\langle p, \alpha B_1 \ldots B_k\beta, 1\rangle, push, l, u_1^R\rangle$,
   - if $l = 1$, add
     - $\langle\langle p, B_1 \ldots B_k\beta, 1\rangle, push, 1, u_1^R\rangle \to \langle\langle p, B_1 \ldots B_k\beta, 0\rangle, pop\rangle$,
     - $\langle\langle p, B_1 \ldots B_k\beta, 0\rangle, pop\rangle \to \langle\langle +, B_1 \ldots B_k\beta, 0\rangle, push, 1, \varepsilon\rangle$,
   - if $l \geq 2$, add
     - $\langle\langle p, \alpha B_1 \ldots B_k\beta, 1\rangle, push, l, u_1^R\rangle \to \langle\langle p, \alpha B_1 \ldots B_k\beta, 0\rangle, join, l\rangle$,
     - $\langle\langle p, \alpha B_1 \ldots B_k\beta, 0\rangle, join, l\rangle \to \langle\langle +, \alpha B_1 \ldots B_k\beta, 0\rangle, push, 1, \varepsilon\rangle$.

We have proved that $L(M) = L(G)$, where M is a generalized $(n+1)$*PPDA*. The proof now follows by Lemma 1. □

Let $n \geq 1$. Analogously as in [2, Theorem 3.1.7], we can prove that the language

$$L_n = \{b(a^i b)^{2n-1} : i \geq 1\} \in \mathscr{L}_P^n - \mathscr{L}_P^{n-1}.$$

Lemma 10. *For all* $n \geq 1$, $L_n \in \mathscr{L}^n_{\{1,2,3,4\}}$.

Informally, the automaton has n pushdowns and each but the one of them contains $a^i b a^i$, for some $i \geq 1$. Thus, two symbols a are put to a pushdown – one to the top and one to the bottom. Finally, the symbol b is pushed to the bottom of all $n-1$ pushdowns, i.e., they contain the string $a^i b a^i b$. Obviously, by the operations new and pop, $ba^i b$ can be simulated and compared with the prefix of the input symbol by symbol during the computation. Thus, the automaton has read $ba^i b$, and the content of each of $n-1$ pushdowns is $a^i b a^i b$, i.e., the automaton has accepted the string $ba^i b(a^i b)^{2(n-1)} = b(a^i b)^{2n-1}$.

Proof. If $n = 1$, the proof is trivial; just push $ba^i b$ to the pushdown. Thus, let $n \geq 2$ and $M = (Q, \{a,b\}, R, s)$ be an nPPDA, where $Q = \{0, p, q, r, s, t, f\}$, and R is constructed as follows.

Phase 1.

1. $s \to \langle 0, push, 1, b\rangle$,
2. $\langle 0, push, 1, b\rangle \to \langle 0, pop\rangle$,
3. $\langle 0, pop\rangle \to \langle p, push, 1, b\rangle$,
4. for $2 \leq i < n-1$,

4a. $\langle p, push, i, b\rangle \to \langle p, push, i+1, b\rangle$,

4b. $\langle p, push, n-1, b\rangle \to \langle q, new, 1\rangle$,

Phase 2.

5. $\langle q, new, 1\rangle \to \langle q, push, 1, a\rangle$,
6. $\langle q, push, 1, a\rangle \to \langle q, pop\rangle$,
7. $\langle q, pop\rangle \to \langle s, push, 1, a\rangle$,
8. for $1 \leq i < n$,

8a. $\langle s, push, i, a\rangle \to \langle r, new, i+1\rangle$,

8b. $\langle r, new, i+1\rangle \to \langle r, push, i+1, a\rangle$,

8c. $\langle r, push, i+1, a\rangle \to \langle r, join, i+1\rangle$,

8d. $\langle r, join, i\rangle \to \langle s, push, i, a\rangle$, $i \geq 2$,

8e. $\langle r, join, n\rangle \to \langle q, new, 1\rangle$,

8f. $\langle r, join, n\rangle \to \langle t, new, 1\rangle$,

Phase 3.

9. $\langle t, new, 1\rangle \to \langle t, push, 1, b\rangle$,
10. $\langle t, push, 1, b\rangle \to \langle t, pop\rangle$,
11. $\langle t, pop\rangle \to \langle t, new, 2\rangle$,
12. for $2 \leq i \leq n$,

12a. $\langle t, new, i\rangle \to \langle t, push, i, b\rangle$,

12b. $\langle t, push, i, b\rangle \to \langle t, join, i\rangle$,

12c. $\langle t, join, i\rangle \to \langle t, new, i+1\rangle$,

Phase 4.

13. $\langle t, new, n+1\rangle \to \langle f, pop\rangle$,
14. $\langle f, pop\rangle \to \langle f, pop\rangle$.

Phase 1 reads b from the input and pushes b to $n-1$ pushdowns. Phase 2 repeatedly reads a from the input and pushes a on the top and to the bottom of all $n-1$ pushdowns. Phase 3 reads b from the input and pushes b to the bottom of all $n-1$ pushdowns. Finally, Phase 4 pops all $n-1$ pushdowns. Clearly, ba^ib has been read from the input and each of $n-1$ pushdowns contains $ba^iba^i\$$, where the top of the pushdown is on the right. Thus, we have $L(M) = L_n$. □

Corollary 4. *For all* $n \geq 1$, $\mathscr{L}_P^n \subset \mathscr{L}_{\{1,2,3,4\}}^{n+1}$.

Proof. The inclusion follows from Lemma 9 and the strictness from Lemma 10. □

The following corollary summarizes the power of nPPDAs known so far.

Corollary 5. *For all* $n \geq 1$, $\mathscr{L}_{\{1,2,3,4\}}^n \subseteq \mathscr{L}_P^{n+1} \subset \mathscr{L}_{\{1,2,3,4\}}^{n+2}$.

Proof. It follows immediately from Lemmas 8 and 9, and the previous corollary. □

Analogously, we can prove that for all $n \geq 2$,

$$K_{n+1} = \{a_1^k a_2^k \ldots a_{n+1}^k : k \geq 1\} \in \mathscr{L}_{\{1,2,3,4\}}^n,$$

which proves the following result.

Corollary 6. *For all* $n \geq 2$, $\mathscr{L}_{\{1,2\}}^n \subset \mathscr{L}_{\{1,2,3,4\}}^n$.

Proof. Ibarra [10, Theorem 2.3] proved that $K_{n+1} \notin \mathscr{L}_R^n = \mathscr{L}_{\{1,2\}}^n$. □

Note that by the trick pushing the content of one pushdown to the bottom of the other, we can prove that for all $n \geq 1$, $K_{2n-1} \in \mathscr{L}_{\{1,2,3,4\}}^n$.

4 Conclusion

In this paper, we discussed two variants of pure multi-pushdown automata that perform complete pushdown pops and proved two infinite language hierarchies they characterize with respect to the number of pushdowns. The following theorem summarizes the results of this paper.

Theorem 3. *1.* $\mathscr{L}_{REG} = \mathscr{L}_{\{1,2\}}^1 = \mathscr{L}_{\{1,2,4\}}^1 = \mathscr{L}_{\{1,2,3\}}^1 = \mathscr{L}_{\{1,2,3,4\}}^1$.

2. For all $n \geq 2$, $\mathscr{L}_{\{1,2\}}^n = \mathscr{L}_{\{1,2,4\}}^n \subset \mathscr{L}_{\{1,2,3\}}^n \subseteq \mathscr{L}_{\{1,2,3,4\}}^n$.

3. For all $n \geq 1$, $\mathscr{L}_{\{1,2\}}^n \subset \mathscr{L}_{\{1,2\}}^{n+1}$.

4. For all $n \geq 1$, $\mathscr{L}_{\{1,2,3,4\}}^n \subset \mathscr{L}_{\{1,2,3,4\}}^{n+2}$.

Moreover, note that by Corollary 3.4 in [3], Corollary of Lemma 3.1.5 in [2], and Corollary 5, any language $L \in \bigcup_{n=1}^{\infty} \mathscr{L}^n_{\{1,2,3,4\}}$ over a one-letter alphabet is regular.

On the other hand, this paper does not answer the question of whether the inclusions $\mathscr{L}^n_{\{1,2,3,4\}} \subseteq \mathscr{L}^{n+1}_{\{1,2,3,4\}}$, $n \geq 1$, and $\mathscr{L}^n_{\{1,2,3\}} \subseteq \mathscr{L}^n_{\{1,2,3,4\}}$, $n \geq 2$, are proper or not. However, we conjecture that these inclusions are proper. Furthermore, one of the interesting questions concerning this is whether the language

$$M_n = \{w^n : w \in \{a,b\}^*\}$$

is in $\mathscr{L}^{n-1}_{\{1,2,3,4\}}$, for $n \geq 2$. This is of interest because if M_n is not in $\mathscr{L}^{n-1}_{\{1,2,3,4\}}$, then it implies that

1. $M_n \in \mathscr{L}^n_{\{1,2\}} \cap (\mathscr{L}^n_{\{1,2,3,4\}} - \mathscr{L}^{n-1}_{\{1,2,3,4\}})$ and that
2. $\mathscr{L}^n_{\{1,2\}} \not\subseteq \mathscr{L}^{n-1}_{\{1,2,3,4\}}$,

as it is not hard to see that $M_n \in \mathscr{L}^n_{\{1,2\}}$. Another interesting question is whether the language $K_{n+1} \in \mathscr{L}^n_{\{1,2,3\}}$ because if this is not true, then it implies $\mathscr{L}^n_{\{1,2,3\}} \subset \mathscr{L}^n_{\{1,2,3,4\}}$, for $n \geq 2$. Finally, the following question is of interest from the viewpoint of descriptional complexity: what is the power of pure multi-pushdown automata that perform complete pushdown pops with respect to the number of states?

Acknowledgements

The authors gratefully acknowledge useful suggestions and comments of the anonymous referees.

This work was supported by the Czech Ministry of Education under the Research Plan No. MSM 0021630528 and the Czech Grant Agency project No. GA201/07/0005.

References

[1] Courcelle, B. On jump deterministic pushdown automata. *Math. Systems Theory*, 11:87–109, 1977.

[2] Dassow, J. and Păun, Gh. *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.

[3] Fernau, H. and Holzer, M. Regulated finite index language families collapse. Technical report, University of Tuebingen, 1996.

[4] Fischer, P. C. Multi-tape and infinite-state automata—a survey. *Commun. ACM*, 8(12):799–805, 1965.

[5] Fischer, P. C. and Rosenberg, A. L. Multitape one-way nonwriting automata. *J. Comput. System Sci.*, 2:88–101, 1968.

[6] Ginsburg, S., Greibach, S. A., and Harrison, M. A. One-way stack automata. *J. ACM*, 14:389–418, 1967.

[7] Ginsburg, S. and Spanier, E. Finite-turn pushdown automata. *SIAM J. Control*, 4:429–453, 1968.

[8] Greibach, S. A. Checking automata and one-way stack languages. *J. Comput. System Sci.*, 3:196–217, 1969.

[9] Hopcroft, J. E. and Ullman, J. D. *Formal Languages and Their Relation to Automata*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1969.

[10] Ibarra, O. H. Simple matrix languages. *Inform. and Control*, 17(4):359–394, 1970.

[11] Meduna, A. Simultaneously one-turn two-pushdown automata. *Int. J. Comp. Math.*, 80:679–687, 2003.

[12] Meduna, A. Deep pushdown automata. *Acta Inform.*, 42(8–9):541–552, 2006.

[13] Meduna, A. and Masopust, T. Self-regulating finite automata. *Acta Cybernet.*, 18:135–153, 2007.

[14] Sakarovitch, J. Pushdown automata with terminating languages. *Languages and Automata Symposium, RIMS 421, Kyoto University*, pages 15–29, 1981.

[15] Salomaa, A. *Formal languages*. Academic Press, New York, 1973.

[16] Siromoney, R. On equal matrix languages. *Inform. and Control*, 14:135–151, 1969.

[17] Siromoney, R. Finite-turn checking automata. *J. Comput. System Sci.*, 5:549–559, 1971.

[18] Valiant, L. The equivalence problem for deterministic finite turn pushdown automata. *Inform. and Control*, 81:265–279, 1989.

[19] Wood, D. m-parallel n-right linear simple matrix languages. *Util. Math.*, 8:3–28, 1975.

Received 15th September 2008

Acta Cybernetica **19** (2009) 553–565.

Kleene Revisited by Suschkewitsch

Jean-Marcel Pallo*

Abstract

The aim of this paper is to generalize to nonassociative concatenation the well-known property that the family of left-linear languages is exactly the family of regular languages. For this purpose, we introduce a generalized Kleene star operation.

1 Introduction

It is well-known that the family of left-linear languages is exactly the family of regular languages. This classical result comes from the formal languages theory, which has been developed over free monoids generated by alphabets and equipped with an associative concatenation.

The purpose of this paper is to generalize this statement to the case where the concatenation is no longer associative [13, 14, 15]. The theory of quasigroups [9, 12, 17, 20] is originated from a certain idea of Suschkewitsch [25], which idea we use here for this goal. Given a group $(\mathcal{G}, \star)$, Suschkewitsch observed in 1929 that the proof of Lagrange theorem does not make any use of the associative law: $X \star (Y \star Z) = (X \star Y) \star Z$. This law can be replaced by his more general postulates, $\mathcal{A}$ and $\mathcal{B}$ namely. Postulate $\mathcal{A}$ of [25] can be written as: for all $A, B \in \mathcal{G}$ there is a unique $C \in \mathcal{G}$ such that for all $X \in \mathcal{G}$ we have $(X \star A) \star B = X \star C$. The element C depends upon the elements A and B only and not upon X. If we denote C by $A \circ B$, i.e. $(X \star A) \star B = X \star (A \circ B)$, it is easy to prove that $\circ$ is associative. It has been shown recently in [19] by the author that Postulate $\mathcal{A}$ is a particular case of the concept of relative associativity introduced by Roubaud [21].

Suschkewitsch also considers a special case of Postulate $\mathcal{A}$ which is however more general than the associative law. He states his Postulate $\mathcal{B}$ as: for all $B \in \mathcal{G}$ there is a unique $\widetilde{B} \in \mathcal{G}$ such that for all $X, Y \in \mathcal{G}$ we have $X \star (Y \star B) = (X \star Y) \star \widetilde{B}$. The elements B and $\widetilde{B}$ depend only upon each other. Every B is completely defined by the corresponding $\widetilde{B}$ and conversely. Using Postulate $\mathcal{B}$, it has been shown in [7] that each left-linear language defined with a nonassociative concatenation is a pseudo-regular language, i.e. it can be written by using a generalized Kleene star operation.

*Université de Bourgogne, Département d'Informatique, BP 47870, 21078 DIJON-Cedex, France, E-mail: pallo@u-bourgogne.fr

The converse property (pseudo-regularity implies left-linearity) is established in this paper via a new method. To do this, we are lead to define the inverse $-$ of the operation $\sim$ of Suschkewitsch. Certain coherence problems arise. Then the original groupoid M (i.e. the monoid without associativity) has to be embedded in a larger one. It is absolutely necessary to check that this embedding creates no additional weak associative relation in M, as for example $((xy)(zt))(uv) = (x(yz))((tu)v)$. Afterwards a pseudo-Kleene star operation is defined as a generalization of the standard Kleene star over the monoid. Then we show that the family of left-linear languages defined with a nonassociative concatenation is *exactly* the family of pseudo-regular languages using this new star operation.
Another approach consists of considering trees in place of words, since the notion of a tree strongly reflects nonassociativity. A huge amount of literature relies upon this concept [6].

2 Notations and definitions

Let $V = \{a, b, c, d \ldots\}$ be an alphabet, i.e. a finite nonempty set of letters. We denote by $M = M(V)$ the free groupoid over V equipped with a nonassociative concatenation $\bullet$. The symbol $\bullet$ will be omitted as long as no confusion is possible. To write words of M, parentheses cannot be omitted due to nonassociativity: $x(yz) \neq (xy)z$. By λ we denote the empty word of zero letters. The set of all non-empty words over V is denoted M^+.
To the finite alphabet V, we associate the (infinite) set $\mathcal{V}$ which is the set of letters of V with as a superscript an arbitrary number of symbols $-$ and $\sim$.
We denote by $\mathcal{M} = \mathcal{M}(\mathcal{V})$ the free groupoid over $\mathcal{V}$ equipped with the nonassociative concatenation $\bullet$. Let us define $\mathcal{M}^+ = \mathcal{M}(\mathcal{V}) - \{\lambda\}$. Since $M \subset \mathcal{M}$, a word in M is called a real word. We call a word of $\mathcal{M} - M$ a metaword. Given $w \in \mathcal{M}$, we denote by $|w|$ the length of w, i.e. the number of letters of V in w, each letter is counted as many times it occurs. Let us denote by $\mathcal{M}_n$ the set of words such that $|w| = n$. For example, $w' = ((ab)(cd))(e(bd)) \in \mathcal{M}_7$ is a real word and $w'' = (e(\bar{\tilde{a}}\,\tilde{c}))((a\,\tilde{\tilde{b}}\,)\,\bar{c}) \in \mathcal{M}_6$ is a metaword. The skeleton of a word $w \in \mathcal{M}$ is defined as the real word $sk(w)$ obtained by cancelling all the occurrences of the symbols $-$ and $\sim$. The free word of $w \in \mathcal{M}$ is defined as the word $f(w)$ obtained by cancelling all the occurrences of the symbols $-$, $\sim$, (and). For example, $sk(w'') = (e(ac))((ab)c)$ and $f(w'') = eacabc$.
We call a left (respectively right) word any word where all the open (respectively close) parentheses occur at the beginning (respectively at the end) of the word. For example, $(((((a\,\bar{b})\,\tilde{c})b)a)\,\tilde{c})$ is a left word and $(\tilde{a}\,(b(\bar{a}\,(c(a\,\bar{d})))))$ is a right word.
A weak associative equality of order n is an equation of the form $w' = w''$ where $w', w'' \in \mathcal{M}_n$. For example, $a(b(c(\bar{d}\bar{e}))) = ((ab)c)(\tilde{d}\tilde{e})$ is a weak associative equality of order 5. A weak associative equality is called real if the two words of the

equality are real, i.e. there is no occurrences of $-$ and $\sim$ inside the two words.

Definition 1. *We define the Suschkewitsch algebra $\mathcal{S} = \mathcal{S}(\mathcal{M})$ as the extension of $\mathcal{M}$ with unary operations $-$ and $\sim$ such that the following properties hold for all $P, Q, R \in \mathcal{M}^+$:*

$$\begin{cases} (PQ)R = P(Q\,\bar{R}) \\ P(QR) = (PQ)\,\tilde{R} \\ \bar{\tilde{P}} = \tilde{\bar{P}} = P \\ \overline{PQ} = \bar{P}\bar{Q} \\ \widetilde{PQ} = \tilde{P}\tilde{Q} \end{cases}$$

We will establish later that no real words in $\mathcal{S}$ are forced to collapse by the five previous axioms.

3 Preliminary results

Definition 2. *Let us define the relation $\rightarrow$ on $\mathcal{S}$ as the smallest preordering, invariant with respect to $-$, $\sim$ and $\bullet$, i.e. if $P \rightarrow Q$ then $\bar{P} \rightarrow \bar{Q}$, $\tilde{P} \rightarrow \tilde{Q}$ and $PR \rightarrow QR$, $RP \rightarrow RQ$ for all $R \in \mathcal{M}$, and satisfying for all $P, Q, R \in \mathcal{M}^+$:*

$$P(QR) \rightarrow (PQ)\,\tilde{R}$$

Lemma 1. *We have $AB \rightarrow CD$ iff either (1)*

$$\begin{cases} A \rightarrow C \\ B \rightarrow D \end{cases}$$

or there exists $S \in \mathcal{M}^+$ such that (2)

$$\begin{cases} B \rightarrow S\,\bar{D} \\ AS \rightarrow C \end{cases}$$

Proof. The conditions are obviously sufficient since $AB \rightarrow A(S\,\bar{D}) \rightarrow (AS)\,\tilde{\bar{D}} = (AS)D \rightarrow CD$.
For proving the necessity, let us consider $A, B, C, D \in \mathcal{M}^+$. We define the relation $\prec$ on $\mathcal{M}$: $AB \prec CD$, iff conditions either (1) or (2) are verified.
$\prec$ is reflexive and invariant with respect to $-$,$\sim$ and $\bullet$. Let us prove the transitivity of $\prec$, i.e. if $AB \prec CD$ and $CD \prec EF$ then we have $AB \prec EF$. Among the four

cases to study, we only detail the following one.
If there exist $S, T \in \mathcal{M}^+$ such that

$$\begin{cases} B \to S\,\bar{D} \\ AS \to C \end{cases}$$

and

$$\begin{cases} D \to T\,\bar{F} \\ CT \to E \end{cases}$$

then there exists $U \in \mathcal{M}^+$ such that

$$\begin{cases} B \to U\,\bar{F} \\ AU \to E \end{cases}$$

Indeed, $U = S\,\bar{T}$ verifies: $B \to S\,\bar{D} \to S(\bar{T}\,\bar{\bar{F}}) \to (S\,\bar{T})\,\tilde{\bar{F}} = U\,\bar{F}$ and

$AU = A(S\,\bar{T}) \to (AS)\,\tilde{\bar{T}} = (AS)T \to CT \to E.$ □

Remark 1. The rewrite relation $\to$ is convergent because it is well-known that the rewriting system $x(yz) \rightsquigarrow (xy)z$ is convergent. Thus any word w has a unique normal form denoted by $l(w)$ which is a left word.

Lemma 2. *If $A, B, C, D \in \mathcal{M}^+$, assume that:*
(1) $AB \to CD$,
(2) AB is a real word,
(3) CD is a left word,
(4) there exists $S \in \mathcal{M}^+$ such that $B \to S\,\bar{D}$ and $AS \to C$.
Then we can always choose S as a real word.

Proof. $|D| = 1$ i.e. $D \in \mathcal{V}$ since CD is a left word. $|B| \geq 2$ since $|S| \geq 1$. Thus we can write $B = B_1B_2$. If $|B_2| = 1$, then $B_2 = d \in V$ and we can choose $D = \tilde{d}$ and $S = B_1$ which is a real word. If $|B_2| \geq 2$ and if d is the last letter of B_2, we can write $B_2 = (B_{2,k}(\ldots(B_{2,3}(B_{2,2}(B_{2,1}d)^k$. Then
$B_2 = (B_{2,k}(\ldots(B_{2,3}(B_{2,2}B_{2,1})^{k-1}\,\tilde{\bar{d}}^{k-1}$ and
$B = B_1B_2 = (B_1(B_{2,k}(\ldots(B_{2,3}(B_{2,2}B_{2,1})^k\,\tilde{\bar{d}}^{k}$. We can choose
$S = (B_1(B_{2,k}(\ldots(B_{2,3}(B_{2,2}B_{2,1})^k$ and $D = \tilde{\bar{d}}^{k+1}$. Thus S is a real word. □

Lemma 3. *The following rewriting system $\rightsquigarrow$:*
(R1) $(xy)z \rightsquigarrow x(y\,\bar{z})$

(R2) $\tilde{\bar{x}} \rightsquigarrow x$

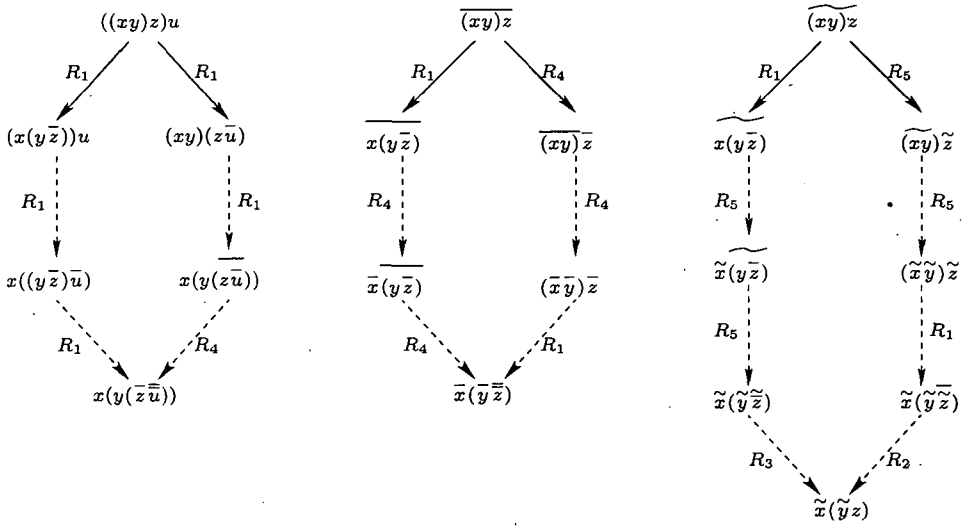


Figure 1: Three convergent critical pairs of $\rightsquigarrow$

(R3) $\overset{\sim}{\overline{x}} \rightsquigarrow x$

(R4) $\overline{xy} \rightsquigarrow \overline{x}\,\overline{y}$

(R5) $\widetilde{xy} \rightsquigarrow \widetilde{x}\,\widetilde{y}$

is convergent. Thus any term t has a unique normal form which is a right word denoted by $r(t)$.

Proof. Termination is easily proved because $\rightsquigarrow$ is included in the Knuth-Bendix ordering. We choose as precedence relation $- \succ \sim \succ \bullet$ and as weights $wgt(-) = wgt(\sim) = 0$ and $wgt(\bullet) = 1$. The superposition of (R1) on (R1), of (R1) on (R4) and of (R1) on (R5) determines three critical pairs which are convergent: see Figure 1. Thus this rewriting system is convergent [1, 10]. Normal forms are right words. □

4 Nonassociativity

Definition 3. *Given $w \in \mathcal{M}_n$, let us call associahedron $\mathcal{AS}_n(w)$ the diagram which is obtained from w by applying all possible $\rightarrow$ and $\overset{-1}{\rightarrow}$ relations.*

See for example the associahedron $\mathcal{AS}_5((x(y(zt)))u)$ in Figure 2.

Theorem 1. *For all n and $w \in \mathcal{M}_n$, the associahedron $\mathcal{AS}_n(w)$ is coherent, i.e. there are no $w', w'' \in \mathcal{AS}_n(w)$ such that $w' \neq w''$ and $sk(w') = sk(w'')$. Therefore*

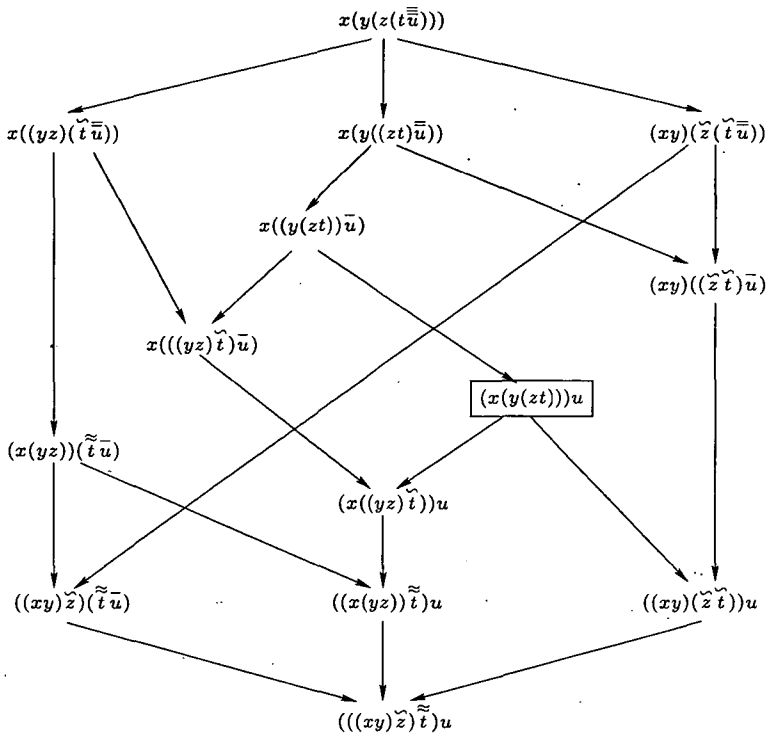


Figure 2: The associahedron $\mathcal{AS}_5((x(y(zt)))u)$

there exist in $\mathcal{AS}_n(w)$ unique words $l(w)$ and $r(w)$ which are respectively left and right words.

Proof. If $n = 4$, the property holds for the five pentagons: see Figure 3. By induction on n, suppose that for $n \geq 5$ and for some $v \in \mathcal{M}_n$ we have in $\mathcal{AS}_n(v)$: $w = AB \rightarrow w' = C'D'$ and $w = AB \rightarrow w'' = C''D''$ with $|w| = |w'| = |w''| = n$, $w' \neq w''$ and $sk(w') = sk(w'')$. Then we obtain $sk(C') = sk(C'')$ and $sk(D') = sk(D'')$. We apply Lemma 1.
If $A \rightarrow C'$, $B \rightarrow D'$ and if $A \rightarrow C''$, $B \rightarrow D''$ then we have $|A| \leq n-1$, $|B| \leq n-1$ and by the induction hypothesis $C' = C''$ and $D' = D''$ since $sk(C') = sk(C'')$ and $sk(D') = sk(D'')$. A contradiction follows since $w' \neq w''$.

If $A \rightarrow C'$, $B \rightarrow D'$, suppose that there exists $S'' \in \mathcal{S}^+$ such that $B \rightarrow S''\ \bar{D}''$ and $AS'' \rightarrow C''$. Therefore we have $|B| = |D'| = |S''| + |D''|$ and $|D'| = |D''|$ because $sk(D') = sk(D'')$. A contradiction follows since $w'' \neq \lambda$.

Now let us assume that there exist $S', S'' \in \mathcal{M}^+$ such that $B \rightarrow S'\ \bar{D}'$, $AS' \rightarrow C'$, $B \rightarrow S''\ \bar{D}''$, $AS'' \rightarrow C''$. If $D' \neq D''$ then we have $l(D') \neq l(D'')$ by induction

because $|D^{'}| \leq n-1$ and $|D^{''}| \leq n-1$. Then $B \to S^{'} \overline{D^{'}} \to l(S^{'})l(\overline{D^{'}})$ and $B \to S^{''} \overline{D^{''}} \to l(S^{''})l(\overline{D^{''}})$. We have $sk(l(S^{'})l(\overline{D^{'}})) = sk(l(S^{''})l(\overline{D^{''}}))$ but $l(S^{'})l(\overline{D^{'}}) \neq l(S^{''})l(\overline{D^{''}})$ because $l(D^{'}) \neq l(D^{''})$. Thus, a contradiction follows. Therefore we obtain $D^{'} = D^{''} = D$, $B \to l(S^{'})D$, $B \to l(S^{''})D$. Since $sk(l(S^{'})D) = sk(l(S^{''})D)$ and $|B| \leq n-1$, we deduce by induction that $l(S^{'})D = l(S^{''})D$, i.e. $l(S^{'}) = l(S^{''})$. Now let us consider the following diagram:

$$C^{'} \leftarrow AS^{'} \to Al(S^{'}) = Al(S^{''}) \leftarrow AS^{''} \to C^{''}.$$

The induction hypothesis can be applied because $|C^{'}| = |C^{''}| \leq n-1$. Then $sk(C^{'}) = sk(C^{''})$ implies $C^{'} = C^{''}$ and a contradiction holds. The existence of $l(w)$ (respectively $r(w)$) is proved using Lemma 1 (respectively Lemma 3). □

Remark 2. The associahedrons $\mathcal{AS}_n(w)$ endowed with the relation $\to$ are lattices for all n and $w \in \mathcal{M}_n$. This is an immediate consequence of the fact that the skeleton of $\mathcal{AS}_n(w)$ is the well-known *n-th* Tamari lattice. Tamari lattices have been extensively studied for algebraic and combinatorial purposes. A number of references on this subject is available in [18].

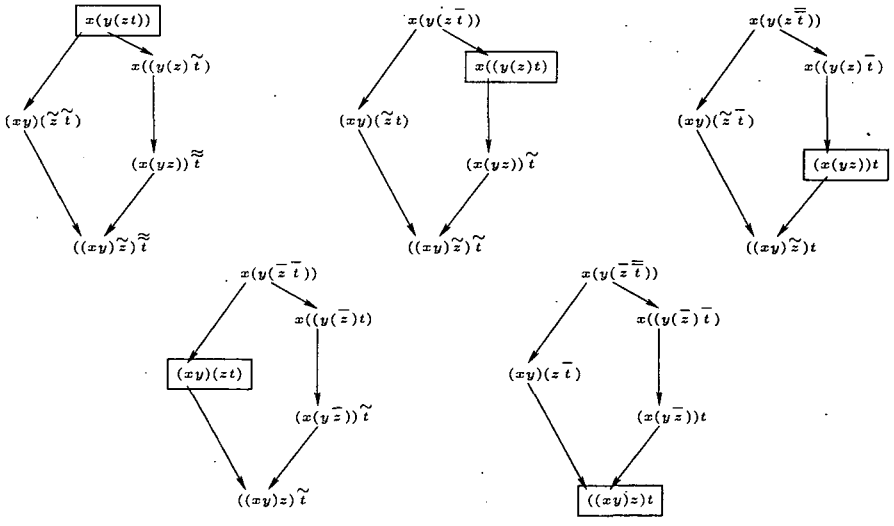


Figure 3: The five associahedrons of size 4

Theorem 2. *The embedding of $\mathcal{M}$ into $\mathcal{S}$ is faithful.*

Proof. Let us prove that the embedding of $\mathcal{M}$ into $\mathcal{S}$ does not create any weak real associativity equality of order n, that is whatever n one cannot find two real words

$w', w'' \in \mathcal{M}_n$ such that $w' = w''$ and $sk(w') \neq sk(w'')$, when applying the five relations of Definition 1.
If $n = 4$, the property holds for the five pentagons: see Figure 3. By induction on n, suppose that for $n \geq 5$ there exist two real words $w' = A'B', w'' = A''B'' \in \mathcal{M}_n$ such that $w' = w''$ with $sk(w') \neq sk(w'')$. Following Lemma 3: $r(w') = r(w'') = v$. In $\mathcal{AS}_n(v)$ there exists a unique left word CD such that $A'B' \rightarrow CD$ and $A''B'' \rightarrow CD$. Since CD is a left word, we have $|D| = 1$. We apply Lemma 1.
If $A' \rightarrow C$, $B' \rightarrow D$, $A'' \rightarrow C$, $B'' \rightarrow D$, then we have $|A'| = |A''| = |C| = n - 1$ and the diagram $A' \rightarrow C \leftarrow A''$ with A' and A'' real, contradicting the induction hypothesis.

If $A' \rightarrow C$, $B' \rightarrow D$, suppose that there exists $S'' \in \mathcal{M}^+$ such that $B'' \rightarrow S''\ \bar{D}$ and $A''S'' \rightarrow C$. Since $|D| = 1$ and B' real, then D is a real word, i.e. $D = d \in V$ and thus $B'' \rightarrow S''\ \bar{d}$. If the arrow $\rightarrow$ is applied to a real word, we can obtain letters with superscript $\sim$ only. Hence, a contradiction follows.

Suppose now that there exist $S', S'' \in \mathcal{M}^+$ such that $B' \rightarrow S'\ \bar{D}$, $A'S' \rightarrow C$, $B'' \rightarrow S''\ \bar{D}$, $A''S'' \rightarrow C$. Following Lemma 2, since B' and B'' are real words, we can choose S' and S'' as real words. Then we have $A'S' \rightarrow C \leftarrow A''S''$ with $A'S'$ and $A''S''$ which are real words. Since $|A'S'| = |A''S''| = |C| = n - 1$, we can apply the inductive hypothesis and we obtain $A'S' = A''S''$. Hence $A' = A''$ and $S' = S'' = S$. The diagram $B' \rightarrow S\ \bar{D} \leftarrow B''$ follows. Since $|B'| \leq n - 1$, the induction hypothesis holds and then $B' = B''$. □

5 Catalan sequences

Definition 4. *If $w \in \mathcal{M}_n$ is a real word, the left word $l(w)$ which is the normal form of w contains only $\sim$ symbols and no $-$ symbols. If $f(w) = x_1x_2 \ldots x_n$, we thus can write $l(w) = ({}^{n-1}x_1x_2)\ \tilde{x}_3^{l_1})\ \tilde{x}_4^{l_2})\ldots)\ \tilde{x}_n^{l_{n-2}})$. The sequence $L(w) = (l_1, l_2, \ldots, l_{n-2})$ is called the left-sequence of $w \in \mathcal{M}_n$.*

Lemma 4. *If $w' \in \mathcal{M}_n$ and $w'' \in \mathcal{M}_m$ with $L(w') = (l'_1, l'_2, \ldots, l'_{n-2})$ and $L(w'') = (l''_1, l''_2, \ldots, l''_{m-2})$, then the left-sequence of the concatenation of w' and w'' is $L(w'w'') = (l'_1, l'_2, \ldots, l'_{n-2}, 0, 1, l''_1 + 1, l''_2 + 1, \ldots, l''_{m-2} + 1)$.*

Proof. If $f(w') = x_1x_2 \ldots x_n$ and $f(w'') = y_1y_2 \ldots y_m$, then we have: $l(w') = ({}^{n-1}x_1x_2)\ \tilde{x}_3^{l'_1})\ldots\ \tilde{x}_n^{l'_{n-2}})$ and $l(w'') = ({}^{m-1}y_1y_2)\ \tilde{x}_3^{l''_1})\ldots\ \tilde{y}_m^{l''_{m-2}})$. Therefore we can write:
$w'w'' = ({}^{n-1}x_1x_2)\ \tilde{x}_3^{l'_1})\ldots\ \tilde{x}_n^{l'_{n-2}})({}^{m-1}y_1y_2)\ \tilde{y}_3^{l''_1})\ldots\ \tilde{y}_m^{l''_{m-2}}) \rightarrow$
$({}^{n}x_1x_2)\ \tilde{x}_3^{l'_1})\ldots\ \tilde{x}_n^{l'_{n-2}})({}^{m-2}y_1y_2)\ \tilde{y}_3^{l''_1})\ldots\ \tilde{y}_m^{l''_{m-2}+1}) \rightarrow$
$({}^{n+1}x_1x_2)\ \tilde{x}_3^{l'_1})\ldots\ \tilde{x}_n^{l'_{n-2}})({}^{m-3}y_1y_2)\ \tilde{y}_3^{l''_1})\ldots\ \widetilde{y_{m-1}}^{l''_{m-3}+1})\ \tilde{y}_m^{l''_{m-2}+1}) \rightarrow \ldots \rightarrow$
$({}^{n+m-2}x_1x_2)\ \tilde{x}_3^{l'_1})\ldots\ \tilde{x}_n^{l'_{n-2}})y_1)\ \tilde{y}_2)\ \tilde{y}_3^{l''_1+1})\ldots\ \tilde{y}_m^{l''_{m-2}+1}) = l(w'w'')$. □

Theorem 3. *An integer sequence* $(l_i)_{1\leq i\leq n-2}$ *is the left-sequence of a word of* $\mathcal{M}_n$ *iff* $l_1 \in \{0,1\}$ *and for all* $i \in [1, n-3]$: $0 \leq l_{i+1} \leq l_i + 1$.

Proof. The proof comes by induction on n using Lemma 4. □

The five left-sequences of the words of $\mathcal{M}_4$ are $\{00, 01, 10, 11, 12\}$. The fourteen left-sequences of the words of $\mathcal{M}_5$ are in lexicographic order:

$$\{000, 001, 010, 011, 012, 100, 101, 110, 111, 112, 120, 121, 122, 123\}$$

The left-sequence characterized just above is exhausted among the 66 Catalan sets in [24, p. 222] where it is denoted by (**u**). The right-sequence of w defined from the right word $r(w)$ has been studied in [7] and appears in [24, p. 222] under the notation (**s**).

6 Rational formal power series

We use the classical notations on formal power series described in [2, 11, 22]. Given a semiring $\mathcal{A}$, we denote by $\mathcal{A}[[\mathcal{S}]]$ the set of formal series

$$s = \sum_{\sigma\in\mathcal{S}} < s, \sigma > \sigma$$

where $< s, \sigma > \in \mathcal{A}$.
The sum of two series is defined in the classical manner. The product $s = s'\,s''$ is defined by $< s, \sigma > = < s', \sigma' > < s'', \sigma'' >$ if $\sigma = \sigma'\sigma''$ and $< s, \sigma > = 0$ otherwise. $s \in \mathcal{A}[[\mathcal{S}]]$ is proper if the coefficient of the right unit λ (i.e. the constant term of s) vanishes: $< s, \lambda > = 0$.
In this case, the series

$$s^\star = \lambda + s + s\,\bar{s} + (s\,\bar{s})\,\bar{s} + ((s\,\bar{s})\,\bar{s})\,\bar{s} + (((s\,\bar{s})\,\bar{s})\,\bar{s})\,\bar{s} + \ldots$$

is defined. We have also

$$s^\star = \lambda + s + s\,\bar{s} + s(\bar{s}\;\bar{\bar{s}}) + s(\bar{s}\,(\;\bar{\bar{s}}\;\;\bar{\bar{\bar{s}}})) + s(\bar{s}\,(\;\bar{\bar{s}}\;\;(\;\bar{\bar{\bar{s}}}\;\;\bar{\bar{\bar{\bar{s}}}}))) + \ldots$$

Definition 5. *We call* $s^\star$ *the pseudo-Kleene star of the series* $s \in \mathcal{A}[[\mathcal{S}]]$.

Lemma 5. *Let* $r, s \in \mathcal{A}[[\mathcal{S}]]$ *with* s *proper. Then the unique solution* u *of the left-linear equation* $u = r + us$ *is the series* $u = rs^\star$.

Proof. One has $s^\star = \lambda + s^\star\ \bar{s}$ whence $rs^\star = r + r(s^\star\ \bar{s})$ and $rs^\star = r + (rs^\star)\ \tilde{s} = r + (rs^\star)s$. Conversely, from $u = r + us$ it follows that $u = r + (r + us)s = r + rs + (us)s = r + rs + u(s\ \bar{s})$ and inductively $u = r(\lambda + s + s\ \bar{s} + (s\ \bar{s})\ \bar{s} + \ldots + ({}^n s\ \bar{s})\ \bar{s})\ldots)\ \bar{s}) + u({}^{n+1}s\ \bar{s})\ \bar{s})\ldots)\ \bar{s})$. Thus going to the limit, one gets $u = rs^\star$ since s is proper. □

7 Kleene theorem

Definition 6. *A formal series is pseudo-rational if it is an element of the smallest subset $Rat[[\mathcal{S}]]$ of $\mathcal{A}[[\mathcal{S}]]$ containing $\mathcal{S}$ and closed for the sum, product and pseudo-Kleene star operation $\star$.*

Definition 7. *A left-linear system of order N with pseudo-rational coefficients is a system of the form*

$$u_i = r_i + \sum_{1 \le j \le N} u_j s_{i,j}$$

with $1 \le i \le N$ where all $r_i, s_{i,j} \in Rat[[\mathcal{S}]]$.

Theorem 4. *The components of the N-tuple solution of a left-linear system with proper pseudo-rational coefficients are pseudo-rational series. Conversely, a pseudo-rational series can be obtained as a component of a N-tuple solution of such a system.*

Proof. The proof is done by induction on N. According to Lemma 5, the solution of $u = r + us$ is $u = rs^\star$ which is a pseudo-rational series since $r, s \in Rat[[\mathcal{S}]]$. In a system of order N, u_N is rationally computed from $u_1, u_2, \ldots, u_{N-1}$ and the induction hypothesis is applied.
Conversely, let us prove that the components which are solutions of left-linear systems with pseudo-rational coefficients verify the conditions of Definition 6. Let us denote by u_1 (respectively u_1') the first component of the N-tuple solution (respectively N'-tuple solution) of a system $\mathbf{S}$ (respectively $\mathbf{S}'$):

$$\mathbf{S} : u_i = r_i + \sum_{1 \le j \le N} u_j s_{i,j}, 1 \le i \le N$$

and

$$\mathbf{S}' : u_i' = r_i' + \sum_{1 \le j \le N'} u_j' s_{i,j}', 1 \le i \le N'$$

where all $r_i, r_i', s_{i,j}, s_{i,j}' \in Rat[[\mathcal{S}]]$.
It is easy to exhibit a system which admits as solution $c_1 u_1 + c_1' u_1'$ with $c_1, c_1' \in \mathcal{A}$. Now, let $\hat{u}_i = u_1' u_i$. Then

$$u_1' u_i = u_1' r_i + \sum_{1 \le j \le N} u_1' (u_j s_{i,j})$$

and

$$u_1' u_i = u_1' r_i + \sum_{1 \le j \le N} (u_1' u_j) \widetilde{s_{i,j}}$$

Thus $\overset{\wedge}{u_1} = u_1' u_1$ is the first component of the N-tuple solution of the system $\overset{\wedge}{\mathbf{S}}$:

$$\overset{\wedge}{\mathbf{S}} : \overset{\wedge}{u_i} = u_1' r_i + \sum_{1 \le j \le N} \overset{\wedge}{u_j} \widetilde{s_{i,j}}, 1 \le i \le N$$

To conclude, let $\overset{\vee}{u_i} = u_1^\star u_i$. Then

$$u_1^\star u_i = u_1^\star r_i + \sum_{1 \le j \le N} u_1^\star (u_j s_{i,j})$$

and

$$u_1^\star u_i = u_1^\star r_i + \sum_{1 \le j \le N} (u_1^\star u_j) \widetilde{s_{i,j}}$$

Thus $\overset{\vee}{u_1} = u_1^\star u_1 = u_1^\star - \lambda$ is the first component of the N-tuple solution of the system $\overset{\vee}{\mathbf{S}}$:

$$\overset{\vee}{\mathbf{S}} : \overset{\vee}{u_i} = u_1^\star r_i + \sum_{1 \le j \le N} \overset{\vee}{u_j} \widetilde{s_{i,j}}, 1 \le i \le N$$

□

8 Conclusion

Theorem 4 characterizes à-la-Kleene pseudo-rational series defined with a nonassociative concatenation. The key point in the proof of Theorem 4 is the fact that $-$ and $\sim$ are mutually inverse operations. The axiom $(xy)z = x(y\ \bar{z})$ allows to factor out r in the solution of the equation $u = r + us$ and therefore to obtain $u = rs^*$ where s^* is defined in terms of $-$. The converse axiom $x(yz) = (xy)\ \tilde{z}$ allows to show that products and stars of solutions of left-linear systems are yet solutions of certain other left-linear systems that can be computed in terms of $\sim$. The embedding of the original groupoid into the Suschkewitsch algebra creates no weak associative relation. It means that this embedding is faithful.

Theorem 4 is a generalization of the famous Kleene theorem, one of the cornerstones of theoretical computer science. See also [8, 23] for linear languages, [3] for clock languages, [26] for ∞-languages, [4] for a Conway-like approach and [5] for binoid languages.

If $\mathcal{A}$ has the additional structure of a ring (i.e. subtraction is allowed), let us define on $\mathcal{A}[[S]]$ the bracket: $[s,t] = s\ \bar{t}\ - t\ \bar{s}$. We can easily verify that the Jacobi identity holds: $[s,[t,u]] + [t,[u,s]] + [u,[s,t]] = 0$. This remark could be the starting point of a later study.

Acknowledgments

I am very indebted to the anonymous referees for proofreading my paper.

References

[1] Bachmair, L. *Canonical Equational Proofs.* Birkhäuser, Boston, 1991.

[2] Berstel, J. and Reutenauer, C. *Rational Series and their Languages.* EATCS Monographs on Theoretical Computer Science, vol. 12, Springer, Berlin, 1988.

[3] Bouyer, B. and Petit, A. A Kleene/Büchi-like theorem for clock languages. *J. Autom. Lang. Comb.,* 7:167–186, 2002.

[4] Ésik, Z. and Kuich, W. On iteration semiring-semimodule pairs. *Semigroup Forum,* 75:129 – 159, 2007.

[5] Gazdag, Z. and Németh, Z.L., A Kleene theorem for binoid languages. Submitted to International Journal of Foundations of Computer Science, 2008.

[6] Gécseg, F. and Steinby, M. Tree languages. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages,* vol. 3: Beyond Words, pages 1–68. Springer, 1997.

[7] Germain, C. and Pallo, J. Langages rationnels définis avec une concaténation non-associative. *Theor. Comput. Sci.,* 233:217–231, 2000.

[8] Germain, C. and Pallo, J. Linear languages with a nonassociative concatenation. *J. Autom. Lang. Comb.,* 7:311–320, 2002.

[9] Hausmann, B.A. and Ore, O.Theory of quasigroups. *Amer. J. Math.,* 59:983–1004, 1937.

[10] Jantzen, M. Basics of Term Rewriting. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages,* vol. 3: Beyond Words, pages 269–313. Springer, 1997.

[11] Kuich, W. Semirings and Formal Power Series: Their Relevance to Formal Languages and Automata. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages,* vol. 1: Word, Language, Grammar, pages 609–677. Springer, 1997.

[12] Kunen, K. Quasigroups, loops and associative laws. *J. Algebra,* 185:194–204, 1996.

[13] Kuz'min, E.N. and Shestakov, I.P. Non-Associative Structures. In Kostrikin, A.L. and Shafarevich, I.L., editors, *Chap. II of Algebra VI, Encyclopaedia of Mathematical Sciences,* vol. 57. Springer, 1995.

[14] Lõhmus, J. Preface to the Special Issue on Nonassociative Algebras. Quasigroups and Applications in Physics. *Acta Applic. Math.,* 50:1–2, 1998.

[15] Lõhmus, J., Paal, E. and Sorgsepp L. About nonassociativity in mathematics and physics. *Acta Applic. Math.,* 50:3–31, 1998.

[16] Mac Lane, S. Natural associativity and commutativity. *Rice Univ. Stud.,* 49:28–46, 1963.

[17] Murdoch, D.C. Quasigroups which satisfy certain generalized associative law. *Amer. J. Math.,* 61:509–522, 1939.

[18] Pallo, J.M. Generating binary trees by Glivenko classes on Tamari lattices. *Inform. Process. Lett.,* 85: 235–238, 2003.

[19] Pallo, J.M. Nonassociativity à la Kleene. In Bozapalidis, S. and Rohanis, G., editors, *Proc. Second Conf. Algebraic Informatics,* Lecture Notes in Computer Science, vol. 4728, pages 260–274. Springer, 2007.

[20] Pflugfelder, H.O. Historical notes on loop theory, Comment. *Math. Univ. Carolinae,* 41:359–370, 2000.

[21] Roubaud, J. La notion d'associativité relative. *Math. Sci. Hum.,* 34:43–59, 1970.

[22] Salomaa, A. Formal Languages and Power Series. In J. Van Leeuwen, editor, *In Handbook of Theoretical Computer Science,* vol. B, Chap. 3., pages 103–132. Elsevier, 1990.

[23] Sempere, J.M. On a class of regular-like expressions for linear languages. *J. Autom. Lang. Comb.,* 5 :343–354, 2000.

[24] Stanley, R.P. *Enumerative Combinatorics.* vol. 2, Cambridge University Press, 1999.

[25] Suschkewitsch, A. On a generalization of the associative law. *Trans. Amer. Math. Soc.,* 31:204–214, 1931.

[26] Wilke, T. An algebraic theory for regular languages of finite and infinite words. *Intern. J. Algebra Comput.,* 3: 447–489, 1993.

Received 22nd September 2008

CONTENTS

Weighted Automata: Theory and Applications **245**
Preface 247
Frank Drewes: MA$^\Gamma$ Learners for Recognizable Tree Languages and Tree Series 249
Zoltán Fülöp and Loránd Muzamel: Weighted Tree-Walking Automata 275
Andreas Gebhardt and Johannes Waldmann: Weighted Automata Define a Hierarchy of Terminating String Rewriting Systems 295
Thomas Hanneforth and Kay-Michael Würzner: Statistical Language Models within the Algebra of Weighted Rational Languages 313
Adam Koprowski and Johannes Waldmann: Max/Plus Tree Automata for Termination of Term Rewriting 357
Dietrich Kuske: Weighted and Unweighted Trace Automata 393
Eleni Mandrali and George Rahonis: Recognizable Tree Series with Discounting 411

Automata and Formal Languages **441**
Preface 443
Thomas Ang and Janusz Brzozowski: Languages Convex with Respect to Binary Relations, and Their Closure Properties 445
F. Blanchet-Sadri, Robert Mercaş, and Geoffrey Scott: Counting Distinct Squares in Partial Words 465
L. Breveglieri, A. Cherubini, C. Nuccio, and E. Rodaro: Alphabetical Satisfiability Problem for Trace Equations 479
Helmut Jürgensen and Ian McQuillan: Homomorphisms Preserving Types of Density 499
Pavel Martyugin: Complexity of Problems Concerning Reset Words for Some Partial Cases of Automata 517
Tomáš Masopust and Alexander Meduna: On Pure Multi-Pushdown Automata that Perform Complete Pushdown Pops 537
Jean-Marcel Pallo: Kleene Revisited by Suschkewitsch 553

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János